Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Compressão de Sequências de DNA usando codificação de vídeo

Jaime Cândido Dourado Alves

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Alexandre Zaghetto

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Homero Luiz Píccolo

Banca examinadora composta por:

Prof. Dr. Alexandre Zaghetto (Orientador) — CIC/UnB
Prof. Dr. Bruno Macchiavello — CIC/UnB
Prof. Dr. Flávio Vidal — CIC/UnB

CIP — Catalogação Internacional na Publicação

Alves, Jaime Cândido Dourado.

Compressão de Sequências de DNA usando codificação de vídeo / Jaime Cândido Dourado Alves. Brasília : UnB, 2014.
173 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2014.

1. Bioinformática, 2. H.264, 3. Conversão de sequências de DNA

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Compressão de Sequências de DNA usando codificação de vídeo

Jaime Cândido Dourado Alves

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Alexandre Zaghetto (Orientador)
CIC/UnB

Prof. Dr. Bruno Macchiavello
CIC/UnB

Prof. Dr. Flávio Vidal
CIC/UnB

Prof. Dr. Homero Luiz Píccolo
Coordenador do Bacharelado em Ciência da Computação

Brasília, 21 de Agosto de 2014

Dedicatória

Dedico primeiramente a Deus força motora espiritual, à meus pais Adelaide e Sílvio por todo o apoio e dedicação, à meu orientador Zaghetto que foi essencial, a Carla que melhor amiga não há e a José Carlos e Michelle.

Agradecimentos

Primeiramente a Deus por ter possibilitado que tudo isso ocorreu. A minha mãe, Adelaide, o porto seguro que me deu força e apoio em toda minha vida acadêmica. Meu pai, Silvio, por todo o incentivo dado. À meu orientador Alexandre Zaghetto, que esteve comigo em toda essa etapa e cujo apoio foi essencial, encontrando a solução onde não conseguia ver e pelas aulas de Processamento de Imagens que abriram as portas para meu gosto pelo processamento de imagens e a visão computacional. Aos outros membros da banca Flávio Vidal e Bruno Macchiavello quem além de excelentes professores, fizeram parte dessa importante etapa. À minha colega Carla, por mostrar um sorriso e alto astral até nas piores horas. Sem esquecer do Marcelo, Leonardo, Bruno Bertasso, Choc (Bruno Pessanha), Wilson (Pedro Ilton), Pedro Cunha, Goiano (cujo apelido venceu o nome),Ricardo, Tales, Estevão, Éder, Felipão, Diogo, Rocha, Igor, Lucas, Tiago, Caio e Zé que me acompanharam nessa jornada e a fizeram especial. E a todos os professores que já tive aula e citar nomes vai ser impossível por acabar esquecendo de algum.

Resumo

Na área da bioinformática, uma importante tarefa é o sequenciamento de DNA, porém isso gera grandes quantidades de dados que precisam ser comprimidos. Na área do processamento de imagens, comprimir um vídeo é essencial, inclusive com taxas de compressão impressionantes. Dessa similaridade surgiu a hipótese da possibilidade de compressão de sequências de DNA com métodos de compressão de vídeo. Nesse trabalho foi elaborado uma abordagem prática para validar a hipótese. Transformando a sequência de DNA em um vídeo e usando um codificador H.264 e depois o caminho inverso de decodificar o H.264 e transformar o vídeo em sequência de DNA novamente. Analisando principalmente a perfeição da reconstrução da sequência e quanto foi a taxa de compressão comparado com um compressor padrão. Embora a taxa de compressão não tenha superado o compressor padrão, abriu a possibilidade de comprimir usando técnicas de processamento de imagem.

Palavras-chave: Bioinformática, H.264, Conversão de sequências de DNA

Abstract

In bioinformatics, an important job is DNA sequenciang, but it come with a large amount of data that needs to be compressed. In image prossecing, it is very important to compress video data, moreover it compress with a impressive compression tax. Such similarities give rise a hypothesis of DNA sequence's compression using video compression methods. In this work a practical approach was developed to validate this hypothesis. Transforming DNA sequence in video, run H.264 compression then the inverse process, decode H.264's result and transform video in DNA sequence again. Analysing principally perfect reconstruction of sequence and the compression tax compared to a standard compressor. Although compression tax was not better than standard compression, opened up the possibility of compressing using processing image techniques.

Keywords: Bioinformatics, H.264, Conversion of DNA sequences

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ASCII America Standard Code for Information Interchange - Código Padrão Americano para o Intercâmbio de Informação. 8

DNA Ácido Desoxirribonucleico. 1–3, 5–8, 45, 49, 53, 55, 57, 73, 74

EST Abordagem Estável. 52, 60, 65, 69, 70, 73

ETV5 Abordagem Estável de 5 *bits*. 53, 55, 57, 60, 61, 65, 69, 70, 73

EX3 Abordagem Extensão de 3 *bits*. 55–57, 60, 61, 65, 69, 70, 73

EX4 Abordagem Extensão de 4 *bits*. 56, 57, 61, 65, 69, 70, 73

EX5 Abordagem Extensão de 5 *bits*. 57, 61, 65, 69, 70

NCBI Centro Nacional para Informação Biotecnologica - National Center for Biotechnology Information. 9, 60

PSNR Razão Pico Ruído-Sinal (Peak Signal-to-Noise Ratio). 17, 46, 59–61, 65, 69

RNA Ácido Ribonucleico. 5, 7, 8, 49, 52, 55, 57

SNR Média Quadrática da Relação Sinal-Ruído (Signal-to-Noise Ratio). 17

Capítulo 1

Introdução

A Biologia (de "Bios" que significa "vida" e "logia" que significa "estudo" ou "conhecimento"), embora como ciência coesa e única tenha surgido nos séculos XVIII e XIX, tem as bases fundamentadas nas primeiras ciências que o ser humano desenvolveu como a Medicina, a Botânica ou a Filoterapia, citado pelo HistoryWorld [10]. A Biologia é um vasto campo que possui muitas disciplinas especializadas, algumas tão antigas que remontam à Grecia antiga e outras recentes que só foram possiveis graças ao desenvolvimento tecnológico e de novas ciências tais como a computação.

A Bioinformática, como Hogeweg [12] declarou, é uma dessas disciplinas que só foram possíveis graças ao avanço da tecnologia computacional que, juntamente com técnicas advindas da própria Biologia, poderiam propiciar um entendimento dos modelos de sistemas biológicos como sistemas de processamento de informações e analisar os modelos com os dados reais.

Um desses modelos é a biologia molecular. Desde que T.H Morgan e associados em 1910 [2] começaram a desenvolver as técnicas de mapeamento gênico, até a metodologia da tecnologia de DNA recombinante que impulsionou a engenharia genética com o sequenciamento de proteínas e do ácido desoxirribonucleico (DNA) cujo maior marco foi o projeto genoma humano, cada vez mais espécies de plantas, bactérias e animais também estão sendo mapeados, pois possibilitam a descoberta de doenças ou resistências genéticas [5].

Com a melhoria das máquinas de sequenciamento, o advento de variados projetos de sequenciamento, a biologia molecular logo cresceu em produção de dados e os banco de dados rapidamente alcançaram *terabytes* de tamanho. Desse modo, torna-se essencial a compressão de tal volume de dados [1] para poder armazenar, recuperar e transmitir as informações pela *internet*.

Porém a compressão usando os metodos clássicos, como Lempel-Ziv, raramente obtém resultados satisfatórios, o que torna um terreno fértil para experimentação [4, 13, 24]. Nesse cenário é necessário verificar, e experimentar, outras formas de compressão, para descobrir uma forma melhor.

A compressão de imagem e vídeo é uma área em crescente expansão e melhoria, principalmente devido a industria cinematográfica [21, 23], que ao criar novas técnicas com mais qualidade (4K) ou informação (gravação de 48 quadros/segundo) também força a atualização das técnicas de compressão para acompanhar as evoluções. Poder transformar uma sequência de DNA em vídeo, possível de ser comprimida, permite a essa sequência

estar atualizada constantemente em termos de facilidade de transmissão e armazenamento desses dados.

Assim surgiu a ideia de relacionar esse dois campos ao transformar a sequência de DNA em imagem ou vídeo, fazer uma compressão (que poderia ser com perdas ou sem perdas) nessa esfera [3] e então analisar como se comporta ou fazer alguma pequena transformação anterior nas sequências para então tentar comprimir na área de imagem ou vídeo.

1.1 Objetivos

As hipóteses decorrentes dessa ideia foram:

1. É possível comprimir, uma sequência de DNA com um codificador de vídeo com perdas, e poder recuperar a sequência sem erros;
2. Essa compressão é melhor que os métodos usuais de compressão usadas para compactar sequências de DNA (como o $WinRAR^{®}$, Zip ou outros)

1.1.1 Objetivo geral

Conseguir embasamento prático experimental para confirmar ou refutar as hipóteses apresentadas.

1.1.2 Objetivo específico

Descobrir relações possíveis entre o processamento de imagem e a bioinformática no que tange a compressão de dados.

1.2 Organização dos Capítulos

O Capítulo 2 foi baseado em grande parte no livro de Setubal e Meidanis [22]. Que oferecem o nível de teoria biológica que o trabalho necessita. Informações relativas à bioinformática também estão nesse capítulo. O Capítulo 3 tem como base Gonzalez e Woods [6], faz referência principalmente nos conceitos de compressão de imagem e define alguns algoritmos que podem ser úteis ao objetivo do trabalho. O Capítulo 4 tem como base Richardson [20] para conceituar a codificação de predição de movimento. O Capítulo 5 explica como o trabalho foi desenvolvido. Que módulos foram desenvolvidos e como se comportam. O Capítulo 6 mostra os resultados obtidos pelo programa. E o Capítulo 7 mostra a conclusão do trabalho e planos futuros.

Capítulo 2

Biologia Molecular e Bioinformática

Nesse capítulo são apresentadas algumas bases teóricas. Devido a ter pouca relevância nesse trabalho, que visa comprimir uma sequência de DNA, a transcrição (tradução do DNA para aminoácidos) e síntese proteíca (produção de proteína a partir dos aminoácidos) não foram exploradas. Porém alguns aspectos de como as proteínas são codificadas são necessários em caso de transformações possíveis para compressão.

Esse Capítulo foi baseado no livro de Setubal e Meidanis [22]. Os conceitos de proteínas são apresentados na Seção 2.1, os ácidos nucleícos na Seção 2.2, os códons e genes na Seção 2.3 e na Seção 2.4 são relacionados como os arquivos de sequências genéticas devem ser elaborados. Esses conceitos podem levantar novas abordagens para o futuro.

2.1 Proteínas

Proteinas [15, 22] são polímeros constituídos de cadeias de aminoácidos ligados por laços peptídicos, chamados de ligações peptídicas, e com uma composição tridimensional definida. Uma proteína pode ser construida a partir de 20 tipos de aminoácidos listados na tabela 2.1.

As proteínas podem ter 4 tipos de estruturas mostradas na figura 2.1

- Estrutura primária

  Dada pela sequência de aminoácidos ao longo da cadeia polipeptídica. É o nível estrutural mais simples e base para o arranjo espacial da molécula.

- Estrutura secundária

  Ocorre pelo arranjo de aminoácidos próximos entre si e pela capacidade de rotação das ligações do carbono dos aminoácidos e os grupos amina e carboxilo. Tem o formato de hélice, folha ou às vezes laço.

- Estrutura terciária

  Ocorre pelo enrolamento da hélice ou da folha, é estabilizada por pontes de hidrogênio e ligações dissulfureto. Confere atividade biológica às proteínas.

- Estrutura quaternária

As proteínas que tem duas ou mais cadeias polipeptídicas são chamadas de cadeias quaternárias. É estabilizada por pontes de hidrogênio e ligações dissulfureto. Um dos exemplares da estrutura quaternária é a hemoglobina.

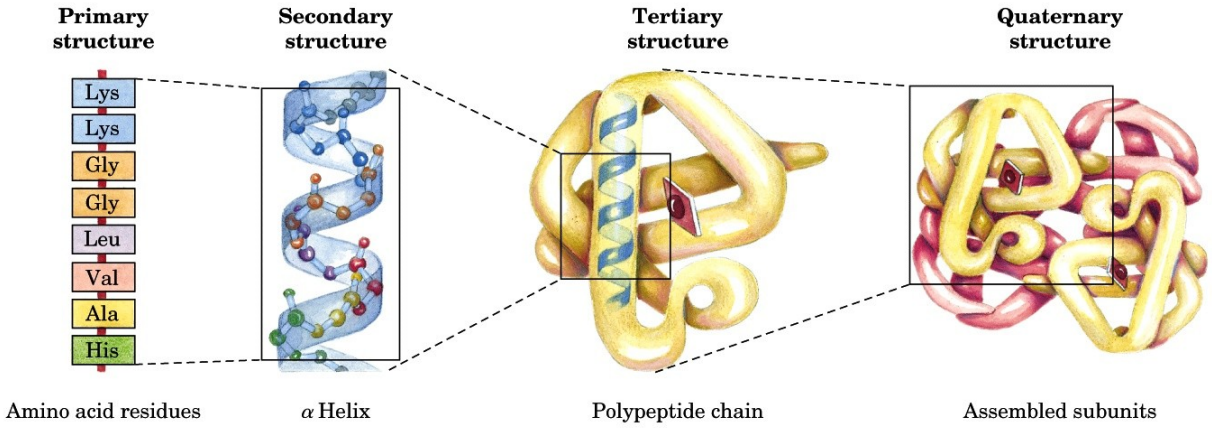


Figura 2.1: Estruturas da proteína

As proteínas estão presentes no corpo de todos os seres vivos (assim como os carboidratos, os lipídeos e os ácidos nucleicos segundo Silva e Nishida [17]) e possuem diversas funções, todas essenciais para o funcionamento do corpo.

- Função Estrutural

  Todas as células são constituidas de proteínas.

- Função Enzimática

  Funcionam como catalizadores, acelerando reações quimicas, ou causando-as.

- Função de transporte

  Transportam ions e pequenas moléculas pelo corpo.

- Função hormonal

  Muitos hormônios do corpo são constituidos de proteínas.

- Função imunológica

  Constituem o sistema imunológico do corpo, se prendem a substâncias estranhas ao corpo e as destroi.

- Função de movimento

  Os músculos são em grande parte constituídos de proteínas. A contração muscular ocorre devido ao deslizamento de filamentos proteícos nas células.

- Função de reserva alimentar

  Proteínas armazenadas em sementes fornecem energia ao organismo durante o desenvolvimento.

Tabela 2.1: Os vinte aminoácidos geralmente encontrados nas proteínas

	Código de uma letra	Código de três letras	Nome
1	A	Ala	Alanina
2	C	Cys ou Cis	Cisteina
3	D	Asp	Aspartato ou Ácido aspártico
4	E	Glu	Glutamato ou Ácido glutâmico
5	F	Phe ou Fen	Fenilalanina
6	G	Gly ou Gli	Glicina ou Glicocola
7	H	His	Histidina
8	I	Ile	Isoleucina
9	K	Lys ou Lis	Lisina
10	L	Leu	Leucina
11	M	Met	Metionina
12	N	Asn	Asparagina
13	P	Pro	Prolina
14	Q	Gln	Glutamina
15	R	Arg	Arginina
16	S	Ser	Serina
17	T	Thr ou The	Treonina
18	V	Val	Valina
19	W	Trp ou Tri	Triptofano
20	Y	Tyr ou Tir	Tirosina

Essas ligações peptídicas são formados pela síntese de um grupo amina (NH_2) com um grupo carboxilia ($COOH$) e a liberação de uma molécula de água. Assim cada proteína é chamada de uma cadeia de polipeptídeos ou de resíduos, pois são resíduos dos aminoácidos originais.

2.2 Ácidos Núcleicos

São moléculas gigantes (macromoléculas) formadas por unidades menores conhecidas como nucleotídeos. São responsáveis pelo armazenamento e transmissão do código genético assim como a transcrição do código genético em proteína. Existem dois tipos: o Ácido Ribonucleico (RNA) e o Ácido Desoxirribonucleico (DNA).

2.2.1 Ácido Desoxirribonucleico

O DNA [14, 22] é uma molécula de duas cadeias antiparalelas em forma de hélice que contém toda informação genética de um indivíduo, como a cor de cabelo, olho, sexo e possibilita a sintese de proteínas. É formado por uma molécula de açúcar (a desoxirribose, na Figura 2.2), um fosfato e uma base nitrogenada, que pode ser uma purina ou uma pirimidina. A combinação do açúcar, fosfato e a base também é conhecida como nucleótido.

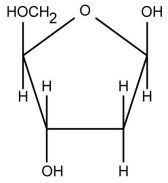


Figura 2.2: Açúcar desoxirribose que compõe o DNA

Essas bases podem ser uma Adenina(purina), Timina(pirimidina), Guanina(purina) ou Citosina(pirimidina) que estão representadas na Figura 2.3.

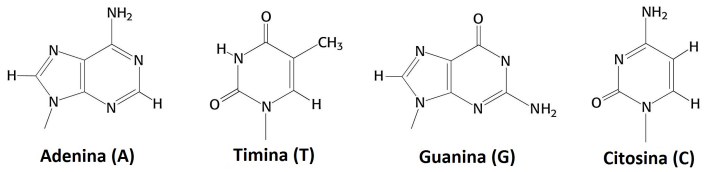


Figura 2.3: Bases que compõem o DNA

O DNA mantém o formato de dupla hélice porque cada base da cadeia é emparelhada com uma base da outra cadeia. A base A sempre é emparelhada com uma base T (Figura 2.4) e uma base C sempre é emparelhada com uma base G (Figura 2.5).

Essas bases são consideradas bases complementares e mantém sua própria orientação, inversas uma da outra, por isso chamadas de antiparalelas. Devido à forte atração das cadeias complementares, nas condições adequadas de salinidade e temperatura as cadeias voltam a se unir. Uma das consequencias imediatas desse fato é que é possível prever a outra cadeia baseada em apenas uma das faixas, assim como possibilita a replicação do DNA.

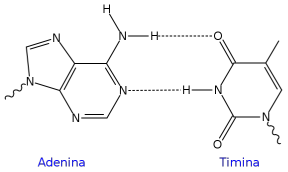


Figura 2.4: Emparelhamento das bases A e T

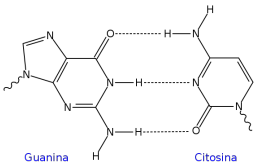


Figura 2.5: Emparelhamento das bases C e G

2.2.2 Ácido Ribonucleico

O RNA [22] é um ácido nucleico formado por um açúcar (a ribose, na Figura 2.6, um fosfato e uma base nitrogenada, que pode ser uma purina ou uma pirimidina.

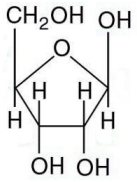


Figura 2.6: Açúcar ribose que compõe o RNA

As diferenças do RNA em relação ao DNA são: a ribose no lugar da desoxirribose; ser uma cadeia simples ao invés da cadeia dupla do DNA; no lugar da Timina é a Uracila (Figura 2.7) que emparelha com a Adenina da mesma forma (Figura 2.8).

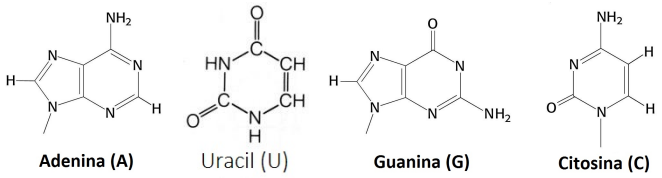


Figura 2.7: Bases que compõem o RNA

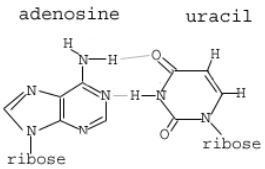


Figura 2.8: Emparelhamento das bases A e U

O RNA não forma dupla hélice como o DNA, a estrutura tridimensional é mais variada que do que no DNA, que enquanto tem a única função de codificar a informação genética, o RNA tem funções diferentes para cada tipo [11, 16].

- RNA mensageiro - mRNA

  Formado no processo de transcrição. Usa uma cadeia de DNA como modelo para o RNA. O mRNA carrega a trinca do código genético do DNA para o ribossomo para transformar o RNA em uma proteína ou cadeia de polipeptídeos.

- RNA ribossômico - rRNA

  Participa na produção de ribossomos que ajudam o mRNA e o RNA transportador para produzir proteínas e cadeias de polipeptídeos.

- RNA transportador - tRNA

  Existem varios tipos de tRNA. Esse RNA transporta os aminoácidos baseado nos codons (baseado na Tabela 2.2), do mRNA, que tem mais afinidade. Formando assim as cadeias de aminoácidos e consequentemente proteínas.

- RNA não-codificante - ncRNA

  São codificador pelos genes do RNA e derivam dos introns do mRNA. Os introns são regiões que não são codificadas na transcrição.

- RNA mensageiro-transportador - tmRNA

  Encontrados em muitas bactérias e plastídeos. Se prendem em proteinas cujo mRNA falta o codon de parada evitando a degradação e a parada dos ribossomos.

2.3 Códons e genes

Os genes são faixas continuas de DNA que servem para construir uma proteína ou alguma molécula de RNA. Como a proteína é uma cadeia de aminoácidos [22], para especificar uma proteína tem que especificar que aminoácidos ela contém. O DNA faz uso de uma tripla de nucleotídeos, essa tripla de nucleotídeos é chamada de *códon*. A Tabela 2.2 mostra a correspondência entre cada tripla e o aminoácido sendo chamado de *código genético*. Por serem as moléculas de RNA que efetivamente sintetizam as proteínas, o quadro usa as bases do RNA, embora o código para a confecção delas está presente no DNA.

Devido a existirem somente 20 aminoácidos e 64 triplas de nucleotídeos possiveis, alguns códons representam o mesmo aminoácido, como por exemplo `CUA` e `UUG` que representarem a Leucina. Porém também existem três codons, `UAG`, `UGA`, `UAA` que não representam nenhum códon, mas são essênciais pois representam o fim do gene, enquanto o códon `AUG` representa o inicio dele.

2.4 Estrutura de dados da bioinformática

As sequências de DNA são armazenadas, no computador, como uma cadeia de caracteres. Como referido por Mount [18], para poder trabalhar com a sequência são precisos duas condições:

- seja um arquivo de texto padronizado como o ASCII (America Standard Code for Information Interchange - Código Padrão Americano para o Intercâmbio de Informação);
- esteja em um formato específico.

Um arquivos de texto ASCII é um texto que possui os caracteres comuns do alfabeto, sem acentos ou cedilha. É importante assegurar que caracteres de controle, que podem ser inseridos por editores profissionais ou formatação, não sejam inseridos no arquivo, sendo aconselhado para tal uso editores mais simples. Esses caracteres especiais podem, além de

Tabela 2.2: Mapeamento de codons para aminoácidos

Primeira Posição	Segunda Posição				Terceira Posição
	G	A	C	U	
G	Gly	Glu	Ala	Val	G
	Gly	Glu	Ala	Val	A
	Gly	Asp	Ala	Val	C
	Gly	Asp	Ala	Val	U
A	Arg	Lys	Thr	Met	G
	Arg	Lys	Thr	Ile	A
	Ser	Asn	Thr	Ile	C
	Ser	Asn	Thr	Ile	U
C	Arg	Gln	Pro	Leu	G
	Arg	Gln	Pro	Leu	A
	Arg	His	Pro	Leu	C
	Arg	His	Pro	Leu	U
U	Trp	STOP	Ser	Leu	G
	STOP	STOP	Ser	Leu	A
	Cys	Tyr	Ser	Phe	C
	Cys	Tyr	Ser	Phe	U

ter dificuldades ao serem transferidos, ser interpretados de outra forma pelos programas que trabalham com as sequências.

Segundo o Comite de Nomenclatura da União Internacional de Bioquímica [19] além das quatro bases padrões (A, C, G, T) são necessários códigos para representar bases ambíguas ou não definidas. Todos os simbolos estão representados na Tabela 2.3.

2.4.1 Formatos de sequências

Existem vários formatos de entrada de sequências como: GenBank, do Laboratório de Biologia Mocular Européia (European Molecular Biology Laboratory - EMBL), SwissProt, FASTA, Grupo de Computação Genética(Genetics Computer Group - GCG), CODATA ou Fundação de Pesquisa Biomédica Nacional/Recurso de Informação de Proteína (National Biomedical Research Foundation/Protein Information Resource - NBRF/PIR), XML entre outras. Mas para o trabalho somente os formatos a seguir são importantes.

2.4.2 GenBank

Formato do banco de dados do Centro Nacional para Informação Biotecnologica - National Center for Biotechnology Information (NCBI)Tem a informação sobre cada entrada da sequência, referências literárias, sobre a função da sequência, sobre a localização dos mRNAs e das regiões de codificação e mutações importantes divida em campos cada uma com um identificador ou uma abreviação dele. A sequência está entre os identificadores "ORIGIN"e "//"e inclui números em cada linha para que as posições sejam mais facil-

Tabela 2.3: Códigos de bases de ácidos nucléicos

Símbolo	Base	Significado
G	G	Guanina
A	A	Adenina
T	T	Tinina
C	C	Citosina
R	A ou G	Purina
Y	C ou T	Pirimidina
M	A ou C	Grupo Amina
K	G ou T	Cetona
S	C ou G	Forte interação
W	A ou T	Fraca interação
H	A, C ou T, mas não G	Letra seguinte à G
B	C, G ou T, mas não A	Letra seguinte à A
V	A,C ou G, mas não T(nem U)	Letra seguinte à T (e à U)
D	A, G ou T, mas não C	Letra seguinte à C
N	A, C, G ou T	Qualquer base

mente localizadas a olho nu. A Figura 2.9 tem uma representação de arquivo no formato GenBank.

```
LOCUS        nome do locus, comprimento de tipo de sequência, classificação do organismo e data de entrada
DEFINITION   descrição de entrada
ACCESSION    número de acesso da fonte original
KEYWORDS     palavras-chave para a referência a esta entrada
SOURCE       organismo fonte de DNA
ORGANISM     descrição do organismo
REFERENCE
COMMENT      função biológica ou informações de banco de dados
FEATURES     informações sobre sequência de posição de base ou série de posições
         source            gama de sequência, organismo fonte
         misc signal       gama de sequencia, tipo de função ou sinal
         mRNA              gama de sequencia, mRNA
         CDs               gama de sequencia, a região de codificação de proteínas
         intron            gama de sequencia, posição de entrada
         mutation          posição sequência, a mudança na sequência por mutação
BASE COUNT    contagem de A, C, G, T e outros símbolos
ORIGIN       texto que indica o início da sequência
       1 gaattcgata aatctctggt ttattgtca gtttatggtt ccaaaatcgc
      51 atatactcac agcataactg tatatacacc caggggcgg aatgaaagcg
//                símbolo do banco de dados para o fim da sequência
```

Figura 2.9: Exemplo de arquivo no formato GenBank - adaptado de Mount [18].

2.4.3 FASTA

O mais conhecido e usado formato de sequência da Bioinformática, principalmente pela simplicidade, pois a sequência não tem espaços, nem números demarcando as posições da sequência. Consiste de 3 partes: 1) uma linha comentário identificada por »"na primeira coluna seguida pelo nome e origem da sequência; 2) a própria sequência; 3) um "*"opcional que pode indicar fim do programa. A Figura 2.10 é uma representação do arquivo no formato FASTA.

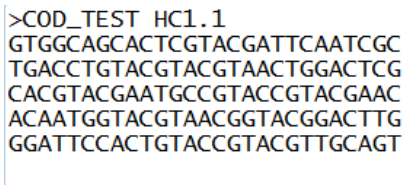


Figura 2.10: Exemplo de arquivo no formato FASTA - adaptado de Mount [18]

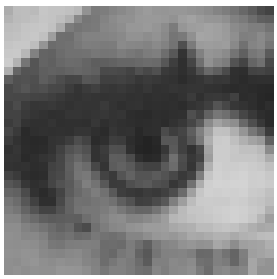
Capítulo 3

Compressão em imagem

Esse Capítulo foi adaptado de Gonzalez e Woods [6] e são explicados alguns conceitos básicos de imagens (na Seção 3.1), na Seção 3.2 são apresentados conceitos teóricos da compressão e na Seção 3.3 são apresentados alguns algoritmos de compressão mais utilizados e mais comuns.quadrosquaqqqqqqqqqwwedec quantidadepixelslllllllllllll

3.1 Definição de imagem

Para as pessoas as imagens de computador podem ser representadas de 3 formas possiveis: uma representação gráfica como uma superfície; um matriz de intensidade visual; e uma matriz como representação numérica, mas para o computador tudo é representado como uma matriz de representação numérica. Na Figura 3.1 é mostrado uma imagem de olho de 31x31 pixels e uma parte da representação numérica do mesmo olho.



172	188	189	186	198	195	195	192	185	146	123
179	187	189	192	197	195	189	177	136	118	121
188	190	196	197	199	193	171	124	116	120	122
191	193	197	199	192	158	111	110	123	130	130
196	199	199	189	149	108	111	113	120	126	125
200	201	183	130	100	98	108	113	113	114	119
204	178	117	85	100	96	104	108	107	101	94
174	101	84	85	95	98	96	100	103	100	96
94	80	82	79	94	87	77	69	70	87	84
72	79	86	77	80	72	71	60	52	59	64
72	71	71	68	67	63	58	53	51	54	52
65	75	60	54	55	57	53	50	53	52	52
67	67	58	53	50	47	48	53	53	51	53
60	67	51	48	48	52	47	48	48	47	55
70	58	42	47	49	48	49	48	53	53	47

Figura 3.1: Representações de imagem: intensidade visual e representação numérica.

Um pixel é chamado de elemento de imagem, elemento pictórico ou pel, ele é a representação numérica que define a intensidade da escala cromática. Essa escala geralmente tem o tamanho de 8 *bits* o que possibilita os valores serem de 1 a 255, embora também exista os de 16 *bits* e os de 32 *bits* esses são mais raros.

Uma imagem monocromática, ou em escala de cinza é uma matriz bidimensional comum, com as linhas representando o eixo y e as colunas representando o eixo x da imagem. A intensidade vai de 0 (preto) a 255 (branco) como mostrado na Figura 3.2.

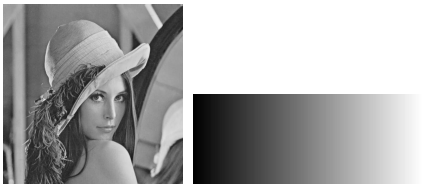


Figura 3.2: Imagem em escala de cinza e a escala de cinza.

Uma imagem colorida é uma matriz tridimensional, além das linhas e colunas, possui um vetor de 3 espaços que representam vermelho (r), verde (g) e azul (b). Cada espaço possui 8 *bits* de intensidade, portanto possui uma faixa de 0 a 255. Como mostrado na Figura 3.3.

Figura 3.3: Imagem colorida e escalas de vermelho, verde e azul.

3.2 Compressão

Como Blelloch [1] salienta a compressão é usada em todos os lugares atualmente e com cada vez mais possibilidades visto que ao mesmo tempo que tecnológias são desenvolvidas os dados gerados também o são. A compressão de dados segundo Folk [7] significa codificar a informação em um arquivo de tal forma que ocupe menos espaço e além disso Gonzalez e Woods [6] salientam que dados são diferentes de informações.

Os dados são os meios pelos quais as informações são transmitidas. Quando varias quantidades de dados podem ser usadas para representar a mesma quantidade de informação, a informação possui *dados redundantes*. Uma forma de calcular a redundância relativa eliminadas na compressão é:

$$R = 1 - \frac{1}{C} \tag{3.1}$$

A Equação 3.1 mostra a redundância relativa de dados eliminados na compressão, R, em termos de bits. E C é a taxa de compressão que está expressa na Equação 3.2. Com b como o dado original e b' é o dado comprimido.

$$C = \frac{b}{b'} \tag{3.2}$$

A compressão de modo geral busca reduzir ou eliminar esses dados redundantes e irrelevantes e podem resultar em dois tipos de algoritmos de compressão: os sem perdas e os com perdas

- Sem perdas - Lossless

  O dado original pode ser reconstruído a partir do dado comprimido sem qualquer diferença. O dado redundante é transformado em uma representação e essa representação é igual ao dado redundante. Tipicamente usada para textos onde cada informação é relevante.

- Com perdas - Lossy

  Com o dado comprimido pode ser reconstruida uma aproximação da informação original. O dado redundante e os dados irrelevantes podem ser eliminado sem grandes perdas para o entendimento, assim impossibilitando fazer o caminho contrario. Tipicamente usada para imagem e som onde algumas partes podem ser omitidas sem perda do significado da mensagem.

Na compressão digital de imagens o b da Equação 3.2 representa o número de bits necessários para representar um arranjo matricial bidimensional. Esse formato, embora perfeito para a visão e interpretação humana, não são adequadas em termos de compactação e são afetadas principalmente por três tipos de redundância:

- Redundância de codificação

  Os códigos de 8 bits utilizados para representar as intensidades do arranjo bidimensional contém mais bits do que o necessário.

- Redundância espacial e temporal

  Como na maioria das intensidades bidimensionais os pixels são correlacionados espacialmente (por semelhança) aos pixels vizinhos e consequentemente neles a informação é replicada. Ou, exclusivamente, em uma sequência de vídeo correlacionados temporalmente aos quadros próximos o que também duplica a informação.

- Informações irrelevantes

  A maioria dos arranjos bidimensionais contém informações que não são facilmente percebiveis pelo sistema visual humano e/ou irrelevantes para a utilização desejada da imagem.

3.2.1 Redundância de codificação

Presuma que em uma imagem MxN no intervalo [0, L-1] seja utilizada a variável discreta r_k para representar as intensidades e $p_r(r_k)$ para a probabilidade de ocorrência sendo:

$$p_k(r_k) = \frac{n_k}{MN} \qquad k = 0, 1, 2, ..., L-1 \tag{3.3}$$

onde L é o número de valores de intensidade, n_k o número de vezes que a k-ésima intensidade aparece e $l_k(r_k)$ o número de *bits* para cada r_k. A média de *bits* necessários para representar cada pixel é:

$$L_med = \sum_{k=0}^{L-1} l(r_k)p_r(r_k) \tag{3.4}$$

Assim o tamanho médio das palavras-código é calculado por meio da soma dos produtos dos números de *bits* utilizados e a probabilidade de ocorrerem. O número total de *bits* da imagem MxN é dado por $M \times N \times L_med$.

Assim a redundância de codificação ocorre quando os códigos atribuídos a um conjunto de eventos não se beneficiam das probabilidades dos mesmos, ou seja, existem intensidades mais possíveis de ocorrerem e outras mais raras. Se beneficia da codificação de tamanho variável. Está quase sempre presente quando as intensidades são representadas usando um código binário natural.

3.2.2 Redundância espacial e temporal

Ocorre quando um arranjo não pode ser comprimido apenas pela codificação de tamanho variável. Os pixels são correlacionados no espaço (em x, y ou ambos) e no tempo (quando faz farte de uma sequência de vídeo).

A maioria das intensidades podem ser preditas a partir das intensidades vizinhas, assim o arranjo deve ser transformado em uma representação mais eficiente, porém "não visual" para as pessoas, como um bloco de 8×8 ser representada por um valor, pares *run-length* ou diferença entre pixels. Essa transformação é conhecida como mapeamento. O mapeamento é reversível se os pixels da imagem original puderem ser reconstruídos sem erros dos dados transformados, senão o mapeamento é irreversível.

3.2.3 Informações irrelevantes

Uma das formas mais simples de comprimir é remover os dados supérfluos do conjunto. E nesse contexto são as informações que o sistema visual humano não consegue ver e/ou que não tenha importância na informação passada pela imagem e possa ser omitida.

Deve-se prestar atenção às estruturas "invisíveis" (intensidades que não foram percebidas, talvez pelas intensidades vizinhas) pois não deixam de ser dados reais e a decisão de preservar ou não as informações depende da aplicação desejada. Em uma aplicação médica, por exemplo, a estrutura poderia ser o indício de algum problema.

Essa redundância é diferente das outras pois a eliminação é possível caso as informações a serem omitidas, não sejam essenciais para o processamento e/ou a utilização da imagem. Como isso resulta em uma perda quantitativa de informações, é chamada de quantização. Esse é um processo irreversível.

3.2.4 Entropia da imagem

Uma das dúvidas que geralmente surgem é a quantidade de *bits* que é realmente necessário para representar as informações da imagem. A teoria da informação proporciona uma estrutura conceitual matemática para isso. A premissa básica diz que a geração de informações pode ser modelada como um processo probabilístico.

Genericamente, dado um evento aleatório E que ocorrá com probabilidade $P(E)$ contém:

$$I(E) = \log \frac{1}{P(E)} = -\log P(E) \tag{3.5}$$

unidades de informação. Se $P(E) = 1$ o evento sempre ocorre, se $P(E) = 0$ nenhuma informação é associada ao evento. A base do logaritmo que define a unidade que vai ser medida. Se a base é 2, a unidade de informação é um *bit*. O que deixa a Equação 3.5 como :

$$I(E) = -\log_2 \frac{1}{2} = 1bit \tag{3.6}$$

assim 1 *bit* é a quantidade de informações quando um dos dois eventos possiveis podem ocorrer.

Dada um conjunto de evento a_j, com $P(a_j)$ como as probabilidades de ocorrerem, é possivel descobrir a quantidade de informação média em unidades base por saída de fonte. Isso é denominada entropia da fonte, cuja formula é:

$$H = -\sum_{j=1}^{J} P(a_j) \log P(a_j) \tag{3.7}$$

Considerando os eventos como os valores de intensidade de uma imagem a Equação 3.7 vira

$$\tilde{H} = -\sum_{k=0}^{L-1} P_r(r_k) \log_2 P_r(r_k) \tag{3.8}$$

A Equação 3.8 é a informação média por saída de intensidade imaginário em *bits*. Assim não é possível codificar os valores de intensidade com menos que $\tilde{H}$ *bits/pixel*.

3.2.5 Fidelidade da imagem

Devido a forma de reduzir informação irrelevante ocorrer pela forma da perda dessas informações, quantificar essa perda fornece uma valiosa ferramenta ao análisar as imagens. Dois critérios podem ser utilizados: 1) *critérios de fidelidade subjetiva* onde a qualidade da imagem descomprimida ocorre pela média de avaliações (com uma escala de valores) de um grupo de pessoas; 2) *critérios de fidelidade objetiva* onde a qualidade da imagem descomprimida é expressa por uma função matemática de entrada e saída.

Como exemplo temos o erro de raiz média quadrática (*root-mean-square* ou rms).

Seja $f(x, y)$ a imagem de entrada e $f'(x, y)$ a imagem comprimida da saída, o erro e

$$e(x, y) = f'(x, y) - f(x, y) \tag{3.9}$$

resulta no erro total entre as imagens de

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f'(x, y) - f(x, y)] \tag{3.10}$$

para imagens com tamanho $M \times N$.

O erro de raiz média quadrática e_{rms} é calculada pela raiz quadrada do erro ao quadrado ao longo de $M \times N$

$$e_{rms} = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[f'(x,y) - f(x,y) \right]^2} \tag{3.11}$$

Se $f'(x,y)$ for a soma da imagem original com um ruído ou erro de transmissão, uma forma de medir é a Média Quadrática da Relação Sinal-Ruído (Signal-to-Noise Ratio) (SNR) dada por

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f'(x,y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[f'(x,y) - f(x,y) \right]^2} \tag{3.12}$$

Se o erro ou ruído de $f'(x,y)$ for derivado de uma compressão uma boa forma de análise é medida pela Razão Pico Ruído-Sinal (Peak Signal-to-Noise Ratio) (PSNR) cuja formula

$$\begin{aligned} MSE &= \tfrac{1}{MN} \textstyle\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[f'(x,y) - f(x,y) \right]^2 \\ PSNR &= 10 \cdot \log_{10} \left(\frac{MAX^2}{MSE} \right) \\ PSNR &= 20 \cdot \log_{10} \left(\frac{MAX}{\sqrt{MSE}} \right) \\ PSNR &= 20 \cdot \log_{10} \left(MAX \right) - 10 \cdot \log_{10} \left(MSE \right) \end{aligned} \tag{3.13}$$

onde MAX é o máximo valor possível para o pixel da imagem. Em 8 *bits* 255.

3.2.6 Modelo de Compressão de imagens

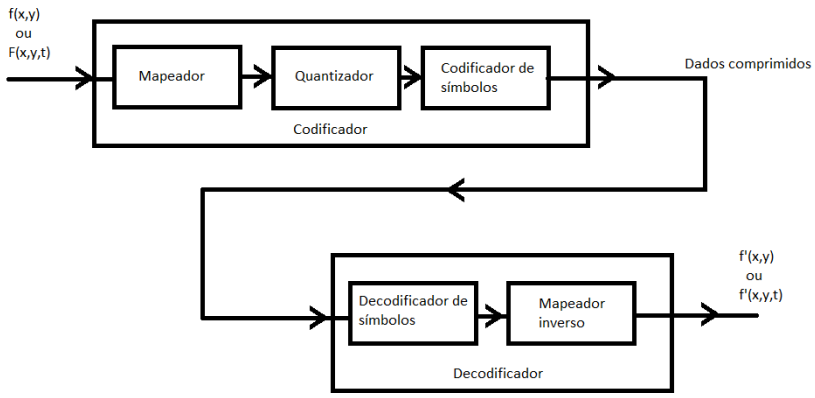


Figura 3.4: Diagrama de um sistema geral de compressão de dados - adaptado de Gonzalez e Woods [6].

Como é mostrado na Figura 3.4, um sistema de compressão de imagens é composto por dois componentes distintos: um codificador e um decodificador. O codificador é encarregado de realizar a compressão e o decodificador a descompressão. Essas operações

podem ser realizadas por aplicativos, uma combinação de hardware e firmware ou através de *codecs* que realizam tanto a codificação quanto a decodificação.

A imagem de entrada $f(x, y)$ (ou $f(x, y, t)$ se for um vídeo) é alimentada no codificador que cria um representação comprimida da imagem. A representação é armazenada para uso futuro, ou pode ser transmitida para armazenamento e utilização por outro dispositivo. No decodificador a representação é usada para gerar uma imagem $f'(x, y)$ (ou $f'(x, y, t)$ se for um vídeo) que pode ou não ser uma copia exata de $f(x, y)$ (ou $f(x, y, t)$). Se a imagem é exata o sistema é classificado como livre de erros, sem perdas ou de preservação de informação, em caso negativo é classificado como compressão com perdas.

Na Figura 3.4 o codificador foi implementado para remover as redundâncias de codificação, redundância espacial e temporal e de informações irrelevantes por meio de três operações diferentes.

No primeiro estagio, o Mapeador, é projetado para reduzir a redundância espacial e temporal. O mapeador transforma $f(x, y)/f(x, y, t)$ em um formato não visual. Essa operação geralmente é reversível e pode reduzir ou não o volume de dados da representação.

O quantizador, na segunda etapa, visa eliminar informações irrelevantes através de um critério de fidelidade predefinido. Isso acaba por reduzir a precisão do mapeamento da etapa anterior. Essa etapa é irreversível e deve ser omitida quando se deseja uma compressão livre de erros.

O codificador de símbolos, na terceira etapa, gera um código de tamanho variável ou fixo para representar e mapear a saída de acordo com o código. Em muitos casos é usado um código de tamanho variável minimizando a redundância de codificação. Essa operação é reversível.

O decodificador é formado de dois componentes: o decodificador de símbolo e um mapeador inverso. Executam na ordem reversa as operações do codificador de simbolos e do mapeamento. Assim primeiro é executado o codificador de símbolos e depois o mapeamento reverso, o quantizador não tem uma operação reversa. Em vídeo os quadros de saída são mantidos em um armazenador e utilizados para reinserir a redundância.

3.3 Algoritmos de compressão clássicos

3.3.1 Código de Huffman

Como definido em Gonzalez [6] é uma das técnicas mais populares. Resulta no menor número possível de símbolos-código ótimo para um valor fixo n, se cumprir certos requisitos e os símbolos-fonte puderem ser codificados um de cada vez. Os símbolos-fontes podem ser as intensidades de uma imagem ou a saída do mapeamento de intensidade.

O código ordena os símbolos baseado nas probabilidades de cada um. Os dois símbolos com menor probabilidade são combinados em uma árvore binária cuja raiz assume o valor da soma das probabilidades das folhas (os símbolos). Rearranja as probabilidades dos símbolos, que incluí a raiz com a nova probabilidade, e combina novamente os dois símbolos com menor probabilidade, repetindo o processo até ter uma árvore binária única com os símbolos originais nas folhas.

A Tabela 3.1 mostra como o procedimento é realizado para um conjunto hipotético. As probabilidades em negrito são as probabilidades que irão se combinar e a célula em azul é a posição que a nova probabilidade vai ocupar.

Tabela 3.1: Redução de fonte pela codificação de Huffman.

Fonte original		Redução de fonte			
Símbolo	Probabilidade	1	2	3	4
a	0,4	0,4	0,4	0,4	0,6
e	0,3	0,3	0,3	**0,3**	0,4
c	0,1	0,1	**0,2**	**0,3**	
b	0,1	**0,1**	**0,1**		
d	**0,06**	**0,1**			
f	**0,04**				

Tabela 3.2: Codificação de fonte pela codificação de Huffman.

Fonte original			Redução de fonte			
Símbolo	Código	Probabilidade	1	2	3	4
a	1	0,4	0,4 1	0,4 1	0,4 1	0,6 0
e	00	0,3	0,3 00	0,3 00	**0,3** 00	0,4 1
c	011	0,1	0,1 011	**0,2** 010	**0,3** 01	
b	0100	0,1	**0,1** 0100	**0,1** 011		
d	01010	**0,06** 01010	**0,1** 0101			
f	01011	**0,04** 01011				

Depois quando essa etapa está concluída, um código binário é atribuido as fontes com menor probabilidade e vai subindo as probabilidades até a de maior. Assim os símbolos que tiverem menor probabilidade terão bits, mais longos e os mais frequêntes terão menos bits. Isso é ilustrado na Tabela 3.2.

O código de Huffman cria o código ótimo para um conjunto de simbolos e probabilidade sob a condição que os símbolos sejam codificados um de cada vez. Depois do código criado a codificação/decodificação ocorre por meio de uma tabela de indexação(*look-up table*) que é instantaneamente decodificável de modo único.

3.3.2 Lempel-Ziv-Welch LZW

Como definido em Gonzalez [6]. Atribui palavras-código de tamanho fixo a sequências de tamanho variável. Uma importante característica da codificação LZW é que não requer conhecimento antecipado da probabilidade de ocorrência dos simbolos que serão codificados. É uma abordagem livre de erros que também lida com as redundâncias espaciais

No inicio do processo é criado um banco de códigos ou dicionário contendo os símbolos-fonte a serem codificados. Exemplo, no campo de imagens monocromáticas (que tem uma faixa de valores de 0 a 255), as primeiras 256 palavras do dicionário são atribuidas os valores de 0 a 255.

A medida que o codificador

A medida que o codificador lê os dados os combina com a entrada anterior, caso não exista no dicionário o novo símbolo é acrescentado nele, se já existir uma entrada com o símbolo no dicionário ele é guardado para ser acrescentado com o próximo símbolo até que

Tabela 3.3: Exemplo de matriz de imagem - copiado de Gonzalez e Woods [6].

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

não exista no dicionário. O processo é melhor entendido, com uma imagem hipotética, observando a Tabela 3.3 como a imagem e a Tabela 3.4 como o dicionário sendo construído.

Na conclusão da codificação nove palavras-código são definidas. Na conclusão do LZW o dicionáio contém 265 palavras, dentre as quais varias sequencias são repetidas.

Claramente o tamanho do dicionário é um importante parâmetro do sistema. Se for pequeno demais, a detecção de sequências será menos próvavel e se for grande demais o tamanho das palavras-código afetará desfavoravelmente o desempenho da compressão.

Um característica notavel é que tanto na hora de comprimir quanto de descomprimir os dicionários, mesmo sendo criados em tempo real, são idênticos.

3.3.3 Run-length

Como definido em Gonzalez [6] elimina a redundância espacial de uma forma simples. Eficaz se o arquivo que estiver sendo comprimido possuir muitas sequências repetidas, chamadas de par de run-length. Cada par de run-lengths define o inicio do byte e quantas vezes aquele byte se repete consecutivamente. Quando há poucas ou nenhuma sequência repetida, a codificação run-length resulta na expanção dos dados.

É particularmente eficaz para imagens binárias, por ter duas intensidades de cores e ser mais provável de se repetirem em sequência.

3.3.4 Codificação aritmética

A códificação aritmética gera códigos sem serem em blocos decodificáveis unicamente, mas sim um único bloco com a mensagem. Na codificação aritmética não existe uma correspondência única entre símbolos-fonte e palavras-código. Uma sequência inteira de símbolos-fonte (ou mensagem) é atribuída a uma única palavra-código aritmética que define um intervalo de números reais entre 0 e 1.

À medida que o número de símbolos na mensagem aumenta, o intervalo para representa-lo diminui e o número de unidades de informação (*bits*) necessários aumentam. Cada símbolo da mensagem reduz o tamanho do intervalo de acordo com a probabilidade de ocorrência .

Usando como hipótese um alfabeto de quatro letras: a_1, a_2, a_3, a_4 e uma mensagem de cinco símbolos $a_1a_2a_3a_3a_4$. É considerado que a mensagem ocupe toda o intervalo [0,1) segundo as probabilidades na Tabela 3.5. O símbolo a_1 é associado ao subintervalo [0 , 0,2). Por ser o primeiro símbolo o intervalo da mensagem é estreitado para [0 , 0,2). O intervalo estreitado é então subdividido de acordo com as probabilidades da Tabela 3.5 e a_2 estreita o [0 , 0,2) em [0,04 , 0,08), a_3 o estreita adicionalmente em [0,056 , 0,072) e assim segue até que a mensagem é estrita até [0,06752 , 0,0688).

Tabela 3.4: Exemplo de codificação LZW .

Sequência atualmente reconhecida	pixel sendo processado	saída processada	posição no dicionário da palavra-código	Entrada do dicionário
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

Tabela 3.5: Exemplo de codificação Aritmética - copiado de Gonzalez e Woods [6].

Símbolo-fonte	Probabilidade	Subintervalo inicial
a_1	0,2	[0,0 ; 0,2)
a_2	0,2	[0,0 ; 0,2)
a_3	0,4	[0,0 ; 0,4)
a_4	0,2	[0,0 ; 0,2)

Na prática, dois fatores fazem a performance de codificação ficar aquém do limite: (1) a adição do indicador de fim da mensagem necessária para separar uma mensagem da outra; e (2) a utilização de precisão aritmética finita.

Com modelos de probabilidade que proporcionam as verdadeiras probabilidades dos símbolos que estão sendo codificados, os codificadores aritméticos se aproximam do ótimo no sentido de minimizar o número médio de símbolos de código necessários.

Uma forma de melhorar a precisão das probabilidades empregadas é utilizar um modelo adaptativo e dependente de contexto. Os *modelos adaptativos* atualizam as probabilidades dos símbolos à medida que eles são codificados ou conhecidos. Os *modelos dependentes do contexto* proporcionam probabilidades baseadas em uma vizinhança predefinida de *pixels* (o contexto) ao redor dos símbolos sendo codificados.

3.3.5 Codificação baseado em símbolos

Na codificação baseada em símbolos ou tokens, a imagem é representada como uma coleção de subimagens de ocorrência frequente, os chamados símbolos. Eles são armazenados em um dicionário de símbolos e a imagem é codificada como um conjunto de triplas $\{(x_1, y_1,t_1), (x_2, y_2,t_2),...\}$ em que o par (x_i,y_i) especifica a posição do simbolo e o *token* t_i, o endereço do símbolo no dicionário. Em aplicações de armazenagem e acesso a documentos, onde os símbolos muitas vezes são mapas de *bits*(*bitmaps*) de caracteres perdidos, poder armazenar os símbolos apenas uma vez, pode comprimir significamente as imagens.

A Figura 3.5 ilustra em A) a imagem que contém a palavra “Banana” que vai ser codificada por símbolos. Em b) temos os *tokens* que foram gerados(“B”, “A” e “N”) e o endereço deles no dicionário de simbolos. Em c) a imagem está mapeada, com a posição em x e y e o endereço do dicionário.

3.3.6 Codificação em plano de bits

A técnica da codificação em plano de bits se baseia no conceito de decompor uma imagem de varios níveis (como uma imagem monocromatica ou colorida) em uma série de imagens binárias que serão comprimidas usando algum método de compressão binária como *run-length* ou baseada em símbolos. A representação na forma do polinomio de base 2 de uma imagem monocromatica é:

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + ... + a_1 2^1 + a_0 2^0 \tag{3.14}$$

Onde m é o número de bits da imagem, no nosso caso estamos usando imagens de 8 *bits*, mas salientando que possam existir imagens com 16 *bits* e 32 *bits*. Assim é possível dividir cada imagem em 8 matrizes bidimensionais de 1 *bit*. O plano com bits de ordem mais baixa é gerado a partir de a_0 de cada pixel e os de plano mais alto são gerados por a_{m-1}. A desvantagem desse método é que pequenas alterações na intensidade podem ter um impacto significativo. Como por exemplo se *pixels* de intensidade 127(01111111) e 128(10000000) forem vizinhos no plano com bits mais altos conterá um *pixel* 0 ao lado de um *pixel* 1. Uma abordagem alternativa é o *código de Gray* de m *bits*. O polimonio(Equação 3.14) pode ser calculado para Gray a partir de

$$g_i = a_i \oplus a_{i+1} \qquad 0 \leq i \leq m - 2 g_{m-1} = a_{m-1} \tag{3.15}$$

Onde $\oplus$ é um OR exclusivo. Assim palavras-código sucessivas vão diferir em apenas um *bit* de posição. O que possibilita que pequenas variações de intensidade tem menos chance de afetar os m planos de *bits*.

A Figura 3.6(a) mostra o plano de *bits* mais significativo (de a_7 e g_7 até a_4 e g_4) e a Figura 3.6(b) mostra o plano de *bits* menos significativo (de a_3 e g_3 até a_0 e g_0). O plano mais significativo é menos complexo e sem muitos detalhes significativos, importantes ou aleatórios. Convem realçar que o código de Gray (a direita) é menos complexo que o plano de *bits* bruto (o a_i).

Na Tabela 3.6 são observadas para cada m a quantidade de *bits* que o plano possui depois de alguma codificação . A imagem sem qualquer codificação possui 678676 *bits*, com a codificação de Gray a taxa é de 1,43:1 (678676/475964) e a codificada normalmente é de 1,35:1 (678676/503916).

3.3.7 Codificação por transformada de blocos

É uma técnica de compressão que divide uma imagem em pequenos blocos não sobrepostos de mesmo tamanho e processa os blocos independentemente usando uma transformada bidimensional. Uma transformada linear e reversível (como a transformada de Fourier) é utilizada para mapear cada bloco em coeficiente da transformada que são então quantizados e codificados. Esse método obtem vantagem da característica de reversivi-

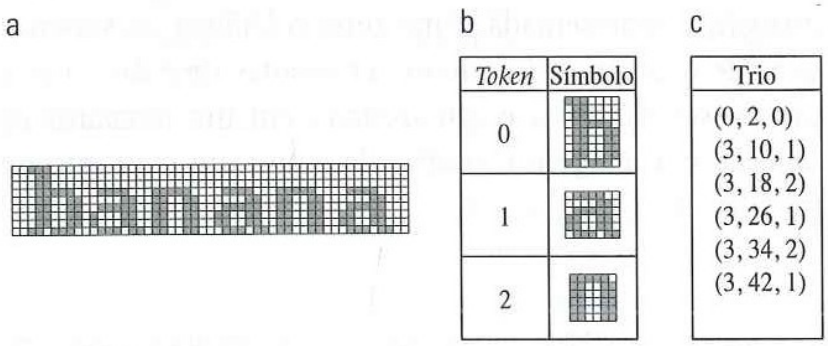


Figura 3.5: Compressão baseada em símbolos - retirado de Gonzalez e Woods [6].

dade para mudar o domínio do problema para outro escopo onde a solução esteja mais acessível (a transformação direta) e posteriormente retornar ao domínio original do problema(a transformação indireta). Muitas imagens possuem um número significativo de coeficientes com pequenas magnetudes que podem ser quantizadas sem grandes prejuízos para a imagem ao voltar ao domínio original.

Um sistema típico de codificação(Figura 3.7) tem como entrada uma imagem $M \times N$ que é subdividida em blocos com tamanho $k \times k$, que são transformadas para gerar arranjos de MN/k^2 subimagens transformadas de tamanho $k \times k$. O objetivo é descorrelacionar os *pixels* de cada subimagem e comprimir o máximo possível as informações em coeficientes de transformada. A quantização elimina ou quantiza seletivamente os coeficientes que carregam menos informação e que consequentemnte tem o menor impacto sobre a qualidade da imagem reconstruida.

Os sistemas de codificação são baseados em uma variedade de transformadas discretas bidimensionais. A escolha de uma transformada para uma aplicação específica depende da quantidade de erros de reconstrução que pode ser tolerada e dos recursos computacionais disponíveis. A compressão é alcançada durante a quantização dos coeficientes transformados, não pela transformada em si.

A transformação geralmente é dada pelas Equação 3.16 para as transformações diretas e pela Equação 3.17 para as indiretas. As transformadas usam um *kernel*, que determina o tipo de transformada calculada, a complexidade geral e o erro de reconstrução do sistema de codificação.

$$T(u,v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} f(x,y) r(x,y,u,v) \tag{3.16}$$

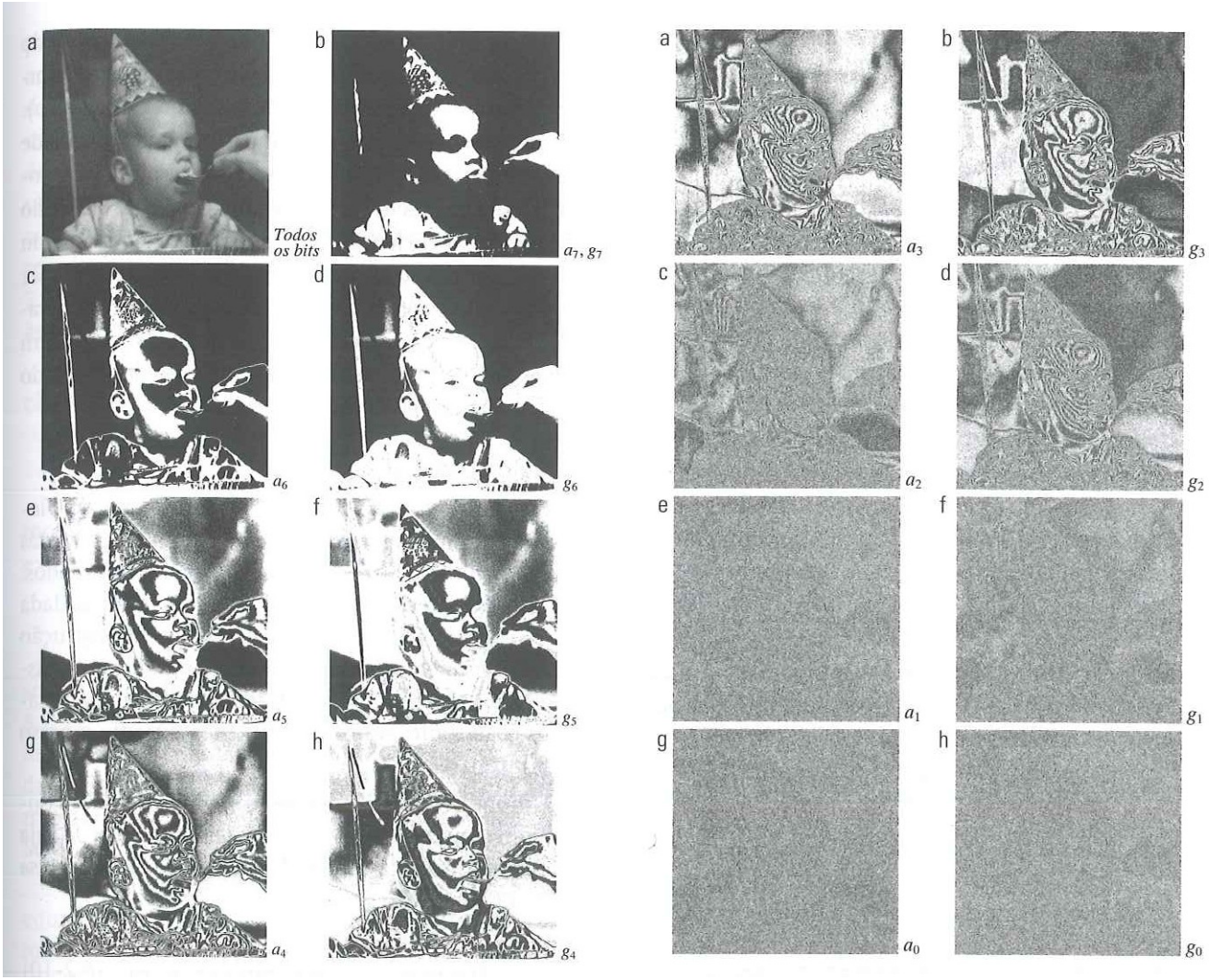
$$f(x,y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u,v) s(x,y,u,v) \tag{3.17}$$

Considere $f(x,y)$ como a subimagem com tamanho $n \times n$, $r(x,y,u,v)$ é o *kernel* da transformação direta e $s(x,y,u,v)$ é o *kernel* da transformação indireta.

A *transformada de Fourier* é a transformada mais conhecida. Usa um *kernel* separável e simétrico que resulta na Equação 3.18 para a transformação direta e a Equação 3.19 para

Tabela 3.6: Resultado da codificação JBIG2 sem perdas do PDF(incluindo cabeçalho) da imagem em 3.6(a)- copiado de Gonzalez e Woods [6].

Coeficiente m	Código binário	Código Gray	Taxa de compressão
7	6999	6999	1,00
6	12791	11024	1,16
5	40104	36914	1,09
4	55911	47415	1,18
3	78915	67787	1,16
2	101535	92630	1,10
1	107909	105286	1,03
0	99753	107909	0,92



(a) Plano de bits mais significativa

(b) Plano de bits menos significativa

Figura 3.6: Plano de bits - retirado de Gonzalez e Woods [6].

a indireta, considere j como $\sqrt{-1}$. Embora o *kernel* seja uma equação computacionalmente complexa, a função poder ser totalmente reconstruída sem perdas de informação.

$$T(u,v) = \sum_{x=0}^{n-1}\sum_{y=0}^{n-1} f(x,y)\exp^{-j2\pi\frac{(ux+vy)}{n}} \tag{3.18}$$

$$f(x,y) = \frac{1}{n^2}\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\exp^{j2\pi\frac{ux+vy}{n}} \tag{3.19}$$

Uma transformada computacionalmente simples e útil na codificação, a *transformada de Walsh-Hadamard* é deduzida dos *kernels* funcionalmente idênticos, ou seja, $r(x,y,u,v) = s(x,y,u,v)$. Como $r(x,y,u,v) = s(x,y,u,v) = \frac{1}{n}(-1)^{\sum_{i=0}^{m-1}|b_i(x)P_i(u)+b_i(y)P_i(v)|}$

as transformações sao dadas pelas Equações 3.20 e 3.21

$$T(u,v) = \frac{1}{n}\sum_{x=0}^{n-1}\sum_{y=0}^{n-1} f(x,y)(-1)^{\sum_{i=0}^{m-1}|b_i(x)P_i(u)+b_i(y)P_i(v)|} \tag{3.20}$$

$$f(x,y) = \frac{1}{n}\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)(-1)^{\sum_{i=0}^{m-1}|b_i(x)P_i(u)+b_i(y)P_i(v)|} \tag{3.21}$$

Considere $n = 2^m$; m a quantidade de *bits* usados; $b_k(z)$ o k-ésimo bit (da direita para aesquerda) da representação de z; assim para $m = 4$ e $z = 11$ $b_0(z) = 1$, $b_1(z) = 1$, $b_2(z) = 0$ e $b_3(z) = 1$; e $P_i(u)$ é definido pela Equação 3.22

$$\begin{aligned} P_0(u) &= b_{m-1}(u) \\ P_1(u) &= b_{m-1}(u) + b_{m-2}(u) \\ P_2(u) &= b_{m-2}(u) + b_{m-3}(u) \\ &\vdots \\ P_{m-1}(u) &= b_1(u) + b_0(u) \end{aligned} \tag{3.22}$$

Os *kernels* de Walsh-Hadamard consiste em alternar 1's positivos e negativos dispostos em um padrão de tabuleiro de xadrez como demonstrado na Figura 3.8 para $n = 4$ ou $m = 2$, com o branco representando $+1$ e o preto o -1.

A *transformada discreta do cosseno*(dct) é uma das mais utilizadas. É composta de vetores ortogonais é muito utilizada pois transfere a maior parte da informação para os primeiros vetores o que facilita a quantização dos valores e o armazenamento (em casos de compressao sem perdas). Como os *kernels* são iguais $r(x,y,u,v) = s(x,y,u,v)$ e o valor é $r(x,y,u,v) = s(x,y,u,v) = \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2y+1)v\pi}{2n}\right]$ e a Equação 3.23 define $\alpha(u)$ e de maneira similar a $\alpha(v)$.

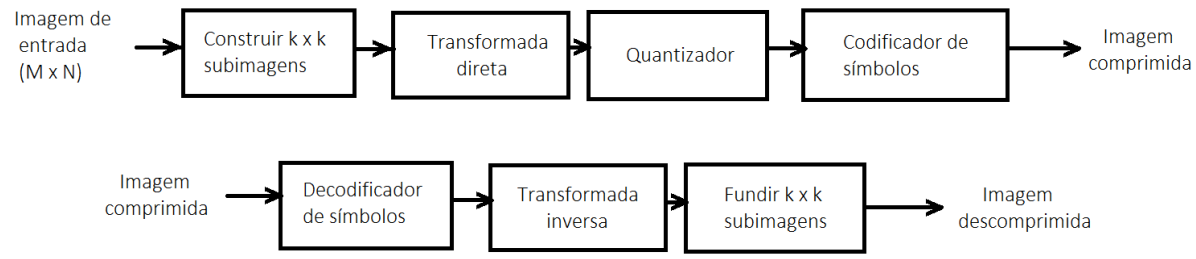


Figura 3.7: Sistema de codificação e descodificação por blocos de transformadas - adaptado de Gonzalez e Woods [6].

$$\alpha(u) = \sqrt{\frac{1}{n}} \qquad \text{para} \quad u = 0$$
$$\alpha(u) = \sqrt{\frac{2}{n}} \qquad \text{para} \quad u = 1, 2, \ldots, n-1 \tag{3.23}$$

Na Figura 3.9 para n=4 temos uma representação dos *kernels* da dct.
Resultando nas Equações 3.24 e 3.25.

$$T(u,v) = \frac{1}{n} \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} f(x,y)\alpha(u)\alpha(v) \cos\left[\frac{(2x+1)u\pi}{2n}\right] \cos\left[\frac{(2y+1)v\pi}{2n}\right] \tag{3.24}$$

$$f(x,y) = \frac{1}{n} \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \alpha(u)\alpha(v) \cos\left[\frac{(2x+1)u\pi}{2n}\right] \cos\left[\frac{(2y+1)v\pi}{2n}\right] \tag{3.25}$$

Na Figura 3.10 é mostrado a aplicação de cada transformada na imagem presente em Figura 3.3 e as imagens de erro decorrente do truncamento de 50% dos coeficientes resultantes.

Desse modo 32 dos coeficientes foram conservados, baseados na máxima magnitude, enquanto 32 foram descartados. Embora a eliminição tenha produzido algum erro quadrático médio, tiveram pouco impacto na qualidade da imagem. O valor de cada erro quadrático médio para Fourier, Walsh-Hadamard e Cosseno foi respectivamente de 2,32 ; 1,78; e 1,13.

O DCT mostrou uma capacidade melhor de compressão que a de Fourier e a de Walsh-Hadamard. Isso geralmente se mantém para a maioria das imagens, mas a transformada de Karhunen-Loève (KLT) é a transformada ótima, devido ao fato de minimizar o erro quadrático médio para qualquer imagem e número de coeficientes. Porém a KLT depende de dados para cada subimagem que não é uma tarefa computacionalmente trivial.

Devido a isso a KLT é pouco utilizada e a DCT devido ao melhor custo-benefício entre capacidade de compressão e complexidade computacional é usada como padrão internacional para a codificação por transformada.

3.3.8 Alocação de bits

O processo de quantização e codificação de uma subimagem transformada é chamada de *Alocação de bits*. O erro de reconstrução funciona como uma função da importância relativa dos coeficientes que foram descartados e da precisão utilizada para os coeficientes conservados. Os coeficientes conservados podem ser selecionados com base na variância máxima, chamada de *codificação por zona*, ou na magnitude máxima, chamada de *codificação por limiarização*.

Codificação por zonas

É baseado no conceito da teoria da informação que vê a informação como incerta. Assim as os coeficientes da transformada com variância máxima carregam a maior parte

da informação relevante da imagem e por isso devem ser conservados no processo de amostragem zonal. Esse processo pode ser visto como a multiplicação de cada subimagem transformada pelo elemento correspondente em uma *máscara de zona* que atribui 1 às posições de máxima variância e 0 às outras posições.

Os coeficientes conservados devem ser então quantizados e codificados, de forma que as máscaras representem o número de bits utilizados para codificar cada um. Geralmente são alocados nos coeficientes o mesmo número de *bits* que são normatizados pelo desvio-padrão e quantizados uniformemente. Embora também podem ser distribuido de maneira desigual onde um quantizador é projetado para cada coeficiente.

Codificação por limiarização

É inerentemente adaptativa visto que a posição dos coeficientes conservados em cada subimagem, varia de uma para outra. É frequentemente a abordagem mais utilizada por pela simplicidade computacional.

Existe três formas básicas de limiarizar uma subimagem transformada: (1) um único limiar global para ser aplicado a todas as subimagens, o nível de compressão difere de uma subimagem para outra dependendo de quantos coeficientes excederem o limiar; (2) um limiar diferente para cada subimagem, descarta o mesmo número de coeficientes, permitindo uma taxa de codificação constante e previamente conhecida; (3) varia em função da posição de cada coeficiente dentro da subimagem, a taxa de codificação é variável, porém a limiarização e a quantização podem ser combinadas.

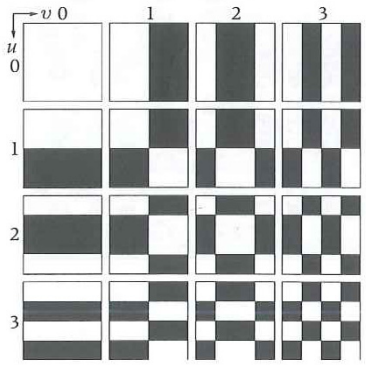


Figura 3.8: Padrão dos *kernels* de Walsh-Hadamard como um tabuleiro de xadrez para n = 4 - retirado de Gonzalez e Woods [6].

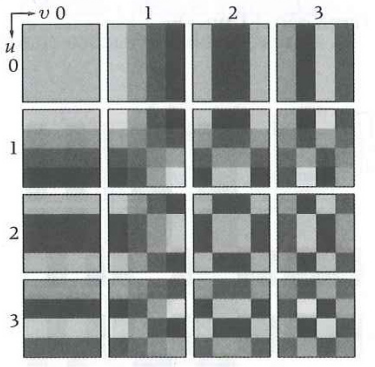


Figura 3.9: Padrão dos *kernels* da DCT para n=4 com blocos de 8x8 - retirado de Gonzalez e Woods [6].

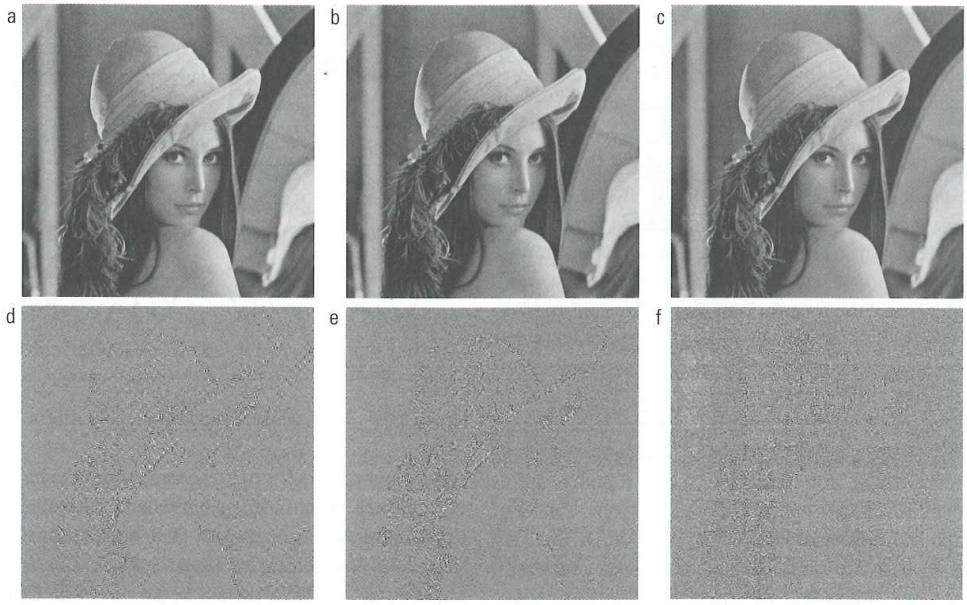


Figura 3.10: Exemplos de transformação usando (a) Fourier (b) Walsh-Hadamard (c) Cosseno (d) a (f) são as imagens de erro correspondentes - retirado de Gonzalez e Woods [6].

Capítulo 4

Compressão em video

4.1 Definição de video

Um cena do mundo real, como na Figura 4.1, é composta de multiplos objetos com formas, texturas, profundidade e iluminação particulares. Embora o brilho e a a cor em um cena do mundo real possa variar um pouco, em essência, tem tonalidade constante.

Figura 4.1: Exemplos de cena do mundo real em video - retirado de Richardson [20].

O vídeo digital é uma representação dessa cena do mundo real, natural ou criada, amostrada no tempo e no espaço para criar um quadro (ou *frame*). O quadro representa uma cena completa de um ponto de visão de um espaço em um periodo de tempo. As amostras são repetidas em intervalos de tempo frequênte (em geral de 25 ou 30 quadros por segundo) para criar um sinal de vídeo como na Figura 4.2. O nosso cérebro ao tentar interpolar as imagens, da imagem anterior com a nova imagem e prevendo como tal mudança ocorreu cria o efeito de animação.

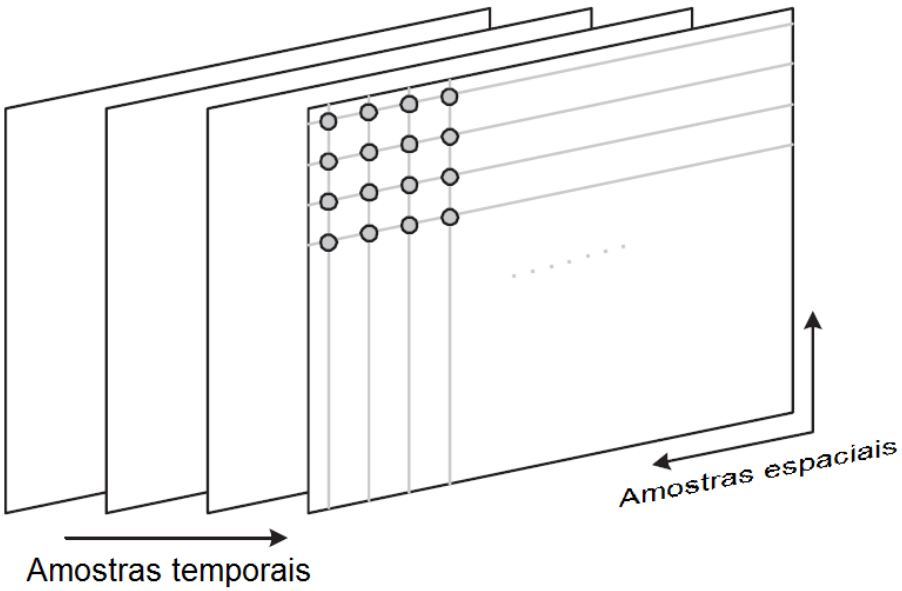


Figura 4.2: Esquema do sinal de video no tempo e espaço - adaptado de Richardson [20].

4.1.1 Amostragem espacial

O quadro ou a imagem amostrada é um conjunto de amostras com valores definidos. O formato mais comum dessas amostras é retangular e são posicionados em uma grade quadrada ou retangular como na Figura 4.3. Em cada intersecção de linha e coluna, o elemento de figura, ou *pixel*, é colocado.

A quantidade dessas amostras influencia na qualidade visual da imagem, uma amostragem grosseira, com poucas amostras, produz imagens com baixa qualidade, enquanto amostragens mais finas, com mais amostras, produz imagens com qualidade melhor.

4.1.2 Amostragem temporal

O efeito de movimento ocorre ao amostrar cada foto do sinal de video em um período de tempo regular. Uma alta taxa de amostragem ou taxa de quadros oferece um movimento mais fluído da cena mas também requer que mais amostras sejam capturadas e armazenadas.

Taxas de quadros abaixo de 10 quadros por segundo podem ser usadas em comunicações com baixa taxa de *bits*, por causa da quantidade de dados pequena, mas o movimento é desajeitado e não natural. Sendo entre 10 a 20 quadros mais comum para comunicações de vídeo com baixa taxa de *bits*, a imagem é um pouco mais suave na movimentação, mas em partes mais rápidas o movimento fica brusco e quebrado. Taxas de amostragem entre 25 a 30 quadros é a norma do *Standard Definition* ou a definição padrão para televisões, que com o interlace (alternância entre linhas pares e ímpares da tela) exibe uma melhora do movimento. Com 50 ou 60 quadros por segundo produz um movimento muito suave ao custo de alta taxa de dados.

Figura 4.3: Grade de pontos - retirado de Richardson [20].

4.1.3 Quadros e campos

O sinal de vídeo pode ser amostrado como uma série de quadros completos, a amostragem progressiva (ou *progressive*), e a amostragem interlaçada (ou *interlaced*, onde apenas uma parte do quadro, as linhas pares ou impares, linha/campo superior e linha/campo inferior, é mostrada por vez, como na Figura 4.4.

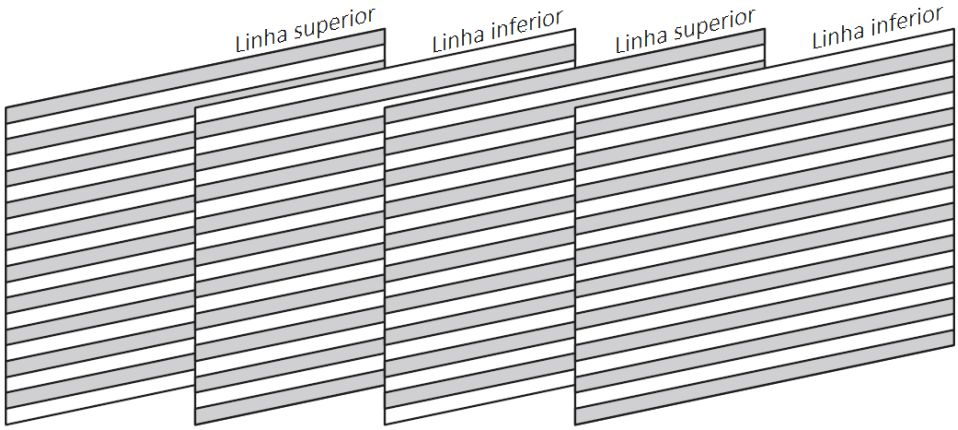


Figura 4.4: Amostragem interlaçada - adaptado de Richardson [20].

4.1.4 Formatos de vídeo

Formatos *Intermediate*

Geralmente é comum capturar e converter quadros de vídeo para um dos formatos de resolução *intermediate* antes de comprimir e transmitir. Escolher a resolução do quadro correta vai depender da aplicação desejada e dos recursos de armazenamento e transmissão disponíveis. O *Common Intermediate Format* (CIF) é o básico desses formatos, o CIF e o QCIF (*Quarter CIF*) são populares para aplicativos de vídeoconferência, 4CIF é mais indicado para televisão e vídeos de DVD.

Na Tabela 4.1 é descrito o formato, o tamanho de resolução e a quantidade de *bits/-quadro* que . Na Figura 4.5 é possível comparar visualmente o tamanho.

Tabela 4.1: Formatos de video *intermediate*.

Formato	Resolução (horiz. x vert.)	Bits por Quadro(4:2:0 e 8 bits por amostra)
Sub-QCIF	128 x 96	147456
Quarter CIF (QCIF)	176 x 144	304128
CIF	352 x 288	1216512
4CIF	704 x 576	4866048

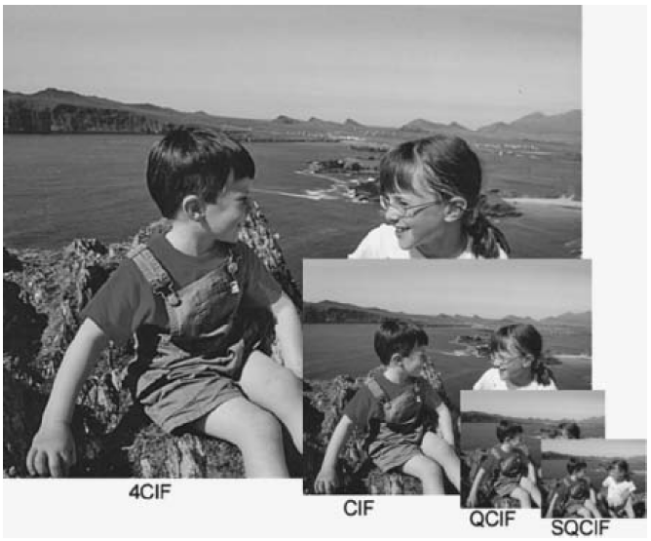


Figura 4.5: Formatos intermedário de video comparativo de resolução - retirado de Richardson [20].

Formato padrão - Standard Definition

Formato altamente utilizado na conversão e transmissão do sinal de video da televisão. O sinal amostrado depende da taxa de quadros, 30Hz para o sinal NTSC e 25Hz para um PAL/SECAM. A Tabela 4.2 teve os dados de luminiscência e crominância omitidos

em razão de ter sido oferecido base teórica necessária. Mas Estão incluidos no calculo da taxa total de *bits* proporcionalmente. Os valores 30 Hz para as linha por quadro e luminiscência e crominância são menores para se adequarem a taxa total de *bits* de 25 Hz e assim ambos os formatos ocuparem a largura de banda.

Tabela 4.2: Formatos de video padrão (standard) - adaptado de Richardson [20].

	30Hz	25Hz
Campos por segundo	60	50
Linhas no quadro completo	525	625
Bits por amostra	8	8
Total da taxa de bits	216Mbps	216Mbps
Linhas ativas	480	576
Amostras de luminiscencia Y ativas por linha	720	720
Amostras de luminiscencia Cb,Cr ativas por linha	360	360

Alta definição - High Definition

Utilizado em DVDs e algumas transmissões de televisão. A Tabela 4.3 mostra a taxa de quadros de uma televisão de alta definição européia (25Hz). Comparando com os valores na Tabela 4.2 para o padrão temos: SD = 10368000 (720x576x25) e HD (720p) = 23040000 (1280x720x25).

Esses valores fazem referência a quantidade de pixels mostrados. Lembrando que cada pixel equivale a 8 *bits*, podemos ter noção de quanta capacidade vai ser necessária para armazenar e transmitir. Demonstrando a necessidade da compressão de videos, principalmente para imagens em alta definição. A Figura 4.6 mostrá o tamanho relativo dos formatos *standard* em relação a *high definition*.

Tabela 4.3: Formatos de video de alta definição (HD).

Formato	Progressivo ou interlaçado	pixels horizontais	pixels verticais	Quadros ou linhas por segundo
720p	Progressivo	1280	720	25 quadros
1080i	Interlaçado	1920	1080	50 quadros
1080p	Progressivo	1920	1080	25 quadros

4.2 Compressão de vídeo

A compressão de vídeo (embora seja um tipo de codificação, nesse trabalho vai ser chamado de codificação de vídeo) é o processo de converter o vídeo digital em um formato adequado para transmissão ou armazenagem, geralmente com a redução do número de

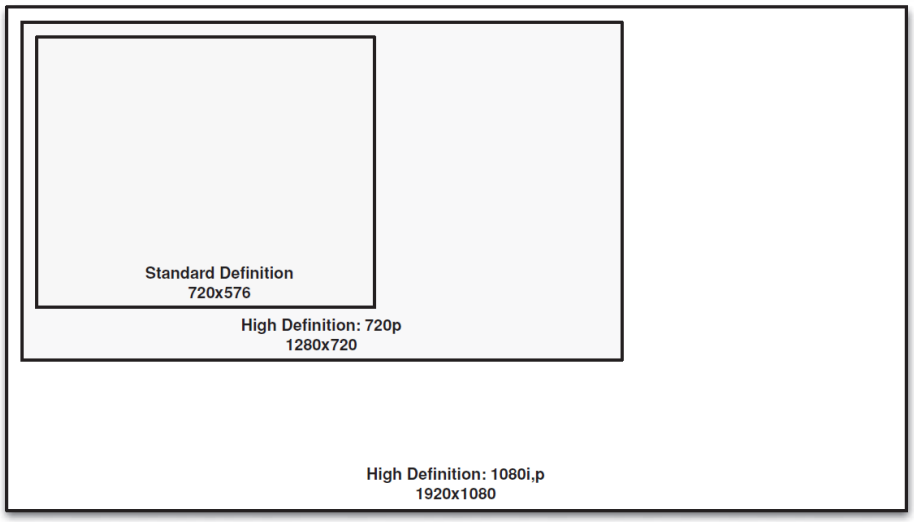


Figura 4.6: Comparativo dos formatos standard e high definition - retirado de Richardson [20].

bits. A compressão envolve um par complementar de sistemas. Um **CO**dificador que converte os dados em um formato com um número reduzido de *bits* antes de armazenar ou transmitir, e o **DEC**odificador que converte a forma comprimida em uma representação do vídeo.

As características das cenas reais que, em geral, são mais relevantes para a codificação de vídeo são: variação de textura na cena, quantidade, forma e cor dos objetos em cena (características espaciais) e movimentação do objeto, mudança na iluminação, movimento da camera ou do ponto de visão (características temporais).

Como salientado no Capítulo 3, a compressão ocorre ao remover a redundância dos dados. Muitos tipos de dados contém uma redundância estatística que permite uma compressão eficiente sem perdas, permitindo que a saída do decodificador seja uma copia exata do dado original. Infelizmente a compressão sem perdas oferece apenas uma quantidade moderada de compressão.

A compressão com perdas é necessária para alcançar uma alta compressão e o dado decodificado não é igual ao dado original, se baseia no fato de remover a redundância subjetiva, elementos que podem ser removidos sem perder significativamente a percepção do usuário.

Muitos métodos de compressão de vídeo exploram tanto a redundância espacial quanto a temporal para conseguir a compressão. Como demonstrado na Figura 4.7. No domínio temporal existe uma alta correlação entre os quadros do vídeo que foram capturadas por volta do mesmo horário. Assim como possui uma alta correlação para os pixels vizinhos.

O H.264, ou *Advanced Video Coding* compartilha muitas características em comum com outros formatos como: MPEG-2 Video, o visual do MPEG-4 e etc. Que usam modelos de predição, blocos de compensação de movimento, transformação, quantização e codificação de entropia.

Padrões internacionais de compressão de vídeo atualmente disponíveis, segundo Gonzalez e Woods [6]:

Figura 4.7: Exemplo de redundância espacial e temporal - retirado de Richardson [20].

- DV

  Vídeo Digital. Um padrão de vídeo projetado para equipamentos e aplicações de produção semi-profissional de vídeo. Os quadros são independentemente comprimidos utilizando uma abordagem baseada em DCT.

- H.261

  Padrão de vídeo conferência bidirecional para linhas ISON (integraded services digital network), suporta imagens 352x288 e 176x144 pixels de resolução. É utilizada uma abordagem baseada em DCT similar ao JPEG com a diferença de usar uma técnica para previsão de quadros e reduzir a redundância temporal.

- H.263

  Uma versão melhorada do H.261 projetada para modens telefônicos comuns.

- H.264

  Uma extensão do H.261-H.263 para videoconferência, streaming de internet e teledifusão. Suporta diferenças de previsão em quadros e usa transformadas de inteiro de tamanho de bloco variável, ao inves de DCT, e codificação aritmética adaptativa ao contexto.

- H.265

  A evolução do H.264 segundo Sullivan e outros [9]. Surgiu para poder acompanhar a evolução dos formatos acima da alta definição, os formatos de ultra definição ($4k \times 2k$, $8k \times 2k$) assim como dispositivos móveis e *tablets*. Usa intra-predição e inter-predição, quantização e transformação de imagens, codificador de entropia e várias unidades e blocos de árvores de codificação inclusive voltadas para processamento paralelo.

- MPEG-1

Um padrão da Motion Pictures Expert Group para aplicações em CD-ROM com vídeo não entrelaçado de até 1,5Mb/s. Similar ao H.261 mas as previsões podem ser baseadas no quadro anterior, no quadro posterior ou em uma interpolação dos dois.

- MPEG-2

  Extensão do MPEG-1 para DVD com taxas de transferência de 15Mb/s com suporte a video entrelaçado e HDTV.

- MPEG-4

  Uma extensão do MPEG-2 que suporta tamanhos variáveis de bloco e diferenciação de previsão em quadros.

- HDV

  High Definition Video. Uma extensão do DV para HDTV que utiliza uma compressão similar ao MPEG-2 incluindo remoção de redundância temporal pela diferenciação de previsão.

- VC-1 WMV9

  Formato mais utilizado na internet. Adotado por HDs, DVDs de alta resolução e Blu-ray. Similar ao padrão H.264/AVC utiliza uma DCT de número inteiro com vários tamanhos de bloco e tabelas de códigos de tamanho variavel dependentes do contexto, mas sem previsão de quadros.

4.2.1 Codec de vídeo

Primeiro o *codec* codifica a imagem ou sequência de video em uma forma comprimida e o decodifica produzindo uma aproximação da fonte. O codec representa a sequência original como um modelo para que possa ser utilizado para a reconstrução do dado. A Figura 4.8 mostra um diagrama com as três unidades principais necessárias para a codificação de video pelo *codec*.

O modelo de predição que recebe a imagem sem compressão e tenta fazer a redução pela similaridade entre quadros vizinhos. O H.264/AVC pode ser mais de um quadro posterior ou anterior para fazer a predição. A saída é um quadro residual que foi criado pela subtração da predição com o quadro original indicando o tipo de predição e como o movimento foi compensado.

O modelo espacial recebe o quadro residual que faz uso das similaridades em amostras locais para reduzir a redundância espacial. No H.264/AVC uma transformada é aplicada no residuo e os coeficientes que resultaram são quantizados. A saída é uma série de coeficientes de transformada quantizados.

O codificador de entropia recebe os parametros de predição do modelo de predição e os coeficientes do modelo espacial, e remove a redundância estatística dos dados, como por exemplo, representando os vetores e coeficientes por códigos binários curtos. O codificador de entropia então produz um *stream* de *bits* comprimidos ou um arquivo que pode ser armazenado ou transmitido. Essa sequência consiste de parametros de predição codificado, coeficientes residuais codificados e cabeçalho de informação.

O decodificador ao receber o *stream* reconstroi o quadro. Os parametros de coeficientes e de predição são codificados pelo decodificador de entropia, depois o modelo espacial é decodificado para reconstruir o quadro residual. O decodificador por fim usa os parametros de predição, juntamente com os *pixels* da imagem decodificados, para criar uma previsão do quadro atual que é reconstruido ao ser adicionado o quadro residual.

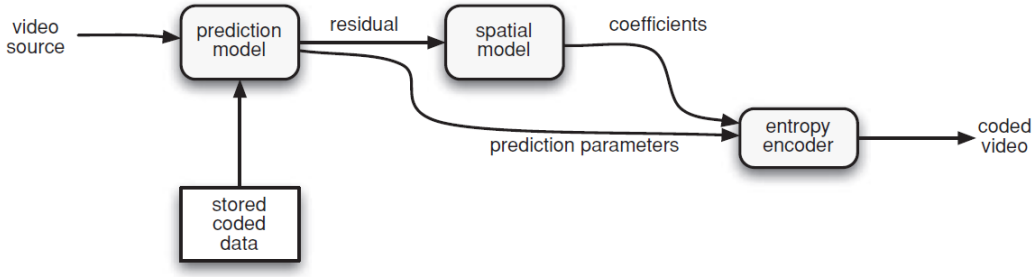


Figura 4.8: Diagrama de bloco do codec - retirado de Richardson [20].

4.2.2 Modelo de predição

Previsão temporal

O objetivo do modelo de predição é reduzir a redundância ao fazer uma predisão do dado e subtraindo-o do dado atual (que já possui) criando o resíduo. A previsão pode ser de quadros já codificados chamados de quadro de referência. A saída é um conjunto de resíduos, quanto menos energia o residuo tiver, mais precisa foi a predição. Esse resíduo também é enviado ao decodificador que de posse do quadro de referência faz a predição do quadro e aplica o resíduo que recebeu.

Na Figura 4.9, o Quadro 1 é usado como referência para prever o Quadro 2. E a diferença entre eles, Quadro 2 - Quadro 1, é o erro de predição. O problema com essa predição é a quantidade de energia que permanece no resíduo, e que seria necessário comprimir para transmitir o dado. Muito dessa energia resídual decorre da mudança gerada pelo movimento entre dois quadros.

Esse movimento inclui regiões sendo descobertas, mudanças de iluminação e movimento de objetos. Com exceção da descoberta de um *background* oculto ou mudança na iluminação, os outros movimentos correspondem a mudanças de posição de um pixel. Sendo possível estimar a trajetória de cada pixel pelo *fluxo óptico*, um campo de vetores com a posição e intensidade de movimento de cada pixel.

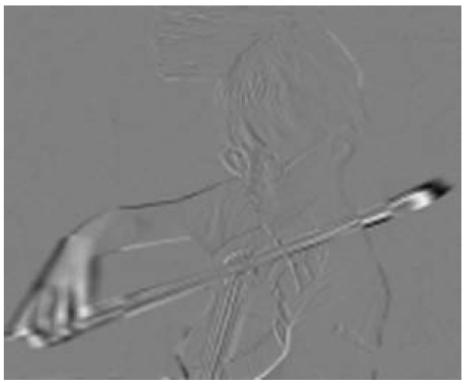
Com um fluxo óptico preciso a predição também o será. Contudo, esse não é um método prático, já que seria computacionalmente intensivo calcular a movimentação de cada pixel, que teria que usar uma abordagem iterativa para cada pixel, e transmiti-lo, pois ocuparia muita largura de banda ao transmitir o vetor.

Estimativa de movimento e compensação de movimento

Um método mais prático e grandemente utilizado para compensar o movimento é movimentar seções retangulares ou blocos do quadro atual. Para estimar cada bloco é utilizado tal procedimento:

Quadro 1

Quadro 2



Quadro 2 - Quadro 1

Figura 4.9: Imagem de exemplo de previção - adaptado de Richardson [20].

- Procurar uma área no quadro de referência com um bloco $M \times N$ similar no quadro atual. Um método de *matching* popular é subtrair a área candidata do bloco $M \times N$ atual, e o candidato com a menor energia residual é escolhida como o melhor par. Esse processo é chamado de *estimação de movimento*.
- O candidato escolhido é a previsão do bloco $M \times N$ atual e é subtraido do bloco para formar um bloco residual $M \times N$, a *compensação de movimento*.
- O bloco residual é codificado, transmitido e a diferença entre a posição do bloco atual e do candidato, *o vetor de movimento*, é transmitido.

O decodificador utiliza o vetor de movimento para re-criar a região de predição. Ao decodificar o bloco residual, acrescentando a previsão, reconstroi uma versão do bloco original.

Embora a maioria dos objetos reais não tenha contornos, são retos para combinar com os blocos retangulares, geralmente se movimentando por posições fracionárias (com a estimação sub-pixel), a predição por blocos é altamente utilizada. Pois é direta, computacionalmente tratável, simétrica para os quadros retangulares e para transformações que usam blocos, como a Transformação Discreta do Cosseno além de prover um modelo razoavelmente efetivo para mudanças temporais.

Em geral a estimativa de movimento usa um bloco de 16×16 no quadro de referência procurando em uma área de busca por um bloco que mais se aproxima do existente no quadro atual. Como pode ser visto na Figura 4.10.

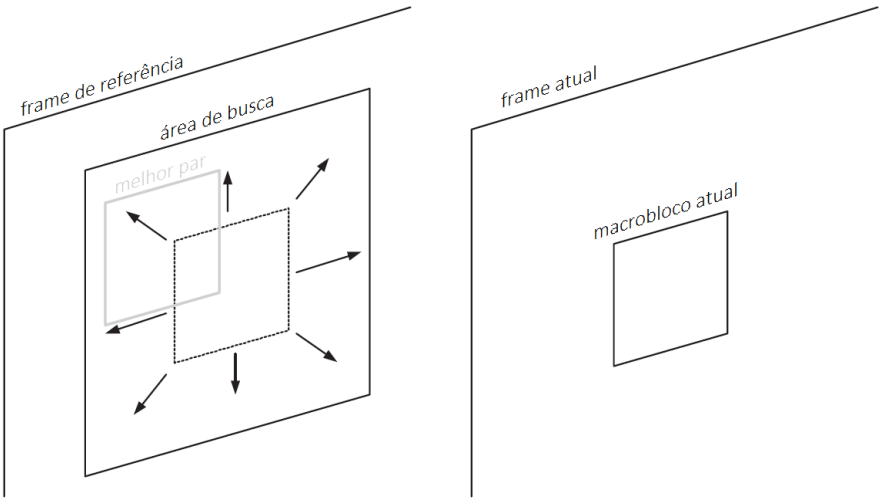


Figura 4.10: Estimativa de movimento com macrobloco - adaptado de Richardson [20].

A região de melhor pareamento no quadro de referência é subtraído do macrobloco atual, produzindo um residuo para o macrobloco que é codificado e transmitido com o vetor de movimento que indica a posição do pareamento em relação a posição do macrobloco atual.

A Figura 4.11 mostra como o tamanho do bloco influência na estimativa de movimento. A energia residual conforme o tamanho do bloco diminui é proporcional. Contudo também

aumenta a complexidade, já que são realizadas mais buscas do macrobloco e aumenta a quantidade de vetores de movimento que deverão ser transmitidos.

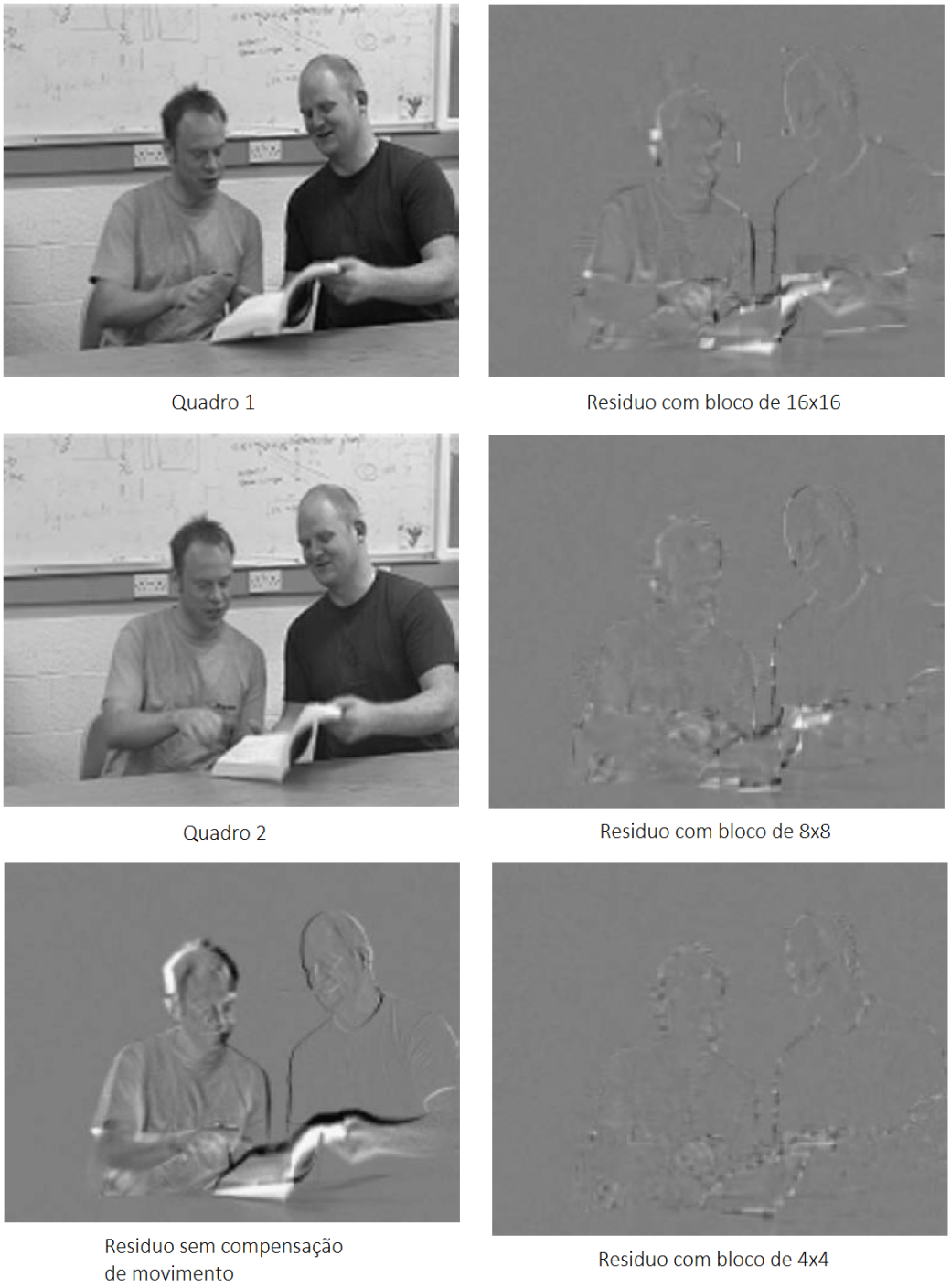


Figura 4.11: Influência do tamanho do macrobloco - adaptado de Richardson [20].

Previsao espacial

Dentro do próprio quadro a predição ocorre a partir das amostras do quadro codificado anteriormente, é o chamada intra-predição. Uma amostra que é codificada no centro da imagem, e assumindo que a codificação é da esquerda para direita e de cima para baixo, as referências possíveis para a previsão são as amostras a esquerda e em cima. Em geral as amostras mais próximas são altamente prováveis de estarem correlacionadas. Uma vez que a predição foi gerada, ela é subtraida do bloco atual para formar um resíduo, nos mesmos moldes que as predições entre quadros. O resíduo é transformado e codificado juntamente com a indicação de como a predição foi feita.

4.2.3 Modelo de imagem

A função do modelo de imagem é descorrelacionar a imagem ou os dados residual para converter em uma forma que pode ser eficientemente comprimida pelo codificador de entropia. Geralmente é baseada em transformar para descorrelacionar e compactar os dados, quantizar para reduzir a precisão dos dados transformados e reordenar para agrupar dados com valores significativos.

Como técnicas temos a: codificação preditiva, codificação por transformada como citada na Subseção 3.3.7.

4.2.4 Codificador de entropia

Converte os símbolos representando as sequências de video em um *bitstream* adequado para transmissão e armazenamento. Usa as técnicas presentes na Seção 3.2.

4.3 H.264

O H.264 *Advanced Video Coding* segundo Richardson [20] é um padrão para a codificação de vídeo, mas também um formato de vídeo codificado, um conjunto de ferramentas para compressão de vídeo e outro estágio na evolução da comunicação de video digital.

O padrão de H.264 foi publicado em 2003 e tem os conceitos baseados no MPEG-2 e MPEG-4 Visual. Oferece uma melhor eficiência em compressão e uma melhor qualidade do vídeo comprimido e grande flexibilidade em comprimir, transmitir e armazenar vídeos comparados com os antecessores. O padrão oferece um formato e uma sintaxe para o video comprimido e um método para decodificar a sintaxe. Não especificando como se deve codificar o vídeo, que é deixado por conta dos fabricantes de codificadores, embora na prática são os passos inversos da decodificação.

Os passos são mostrados na Figura 4.12. A versão decodificada não é igual a original pois o H.264 é uma compressão com perdas.

A predição do H.264 usa intra-predição (dentro do próprio quadro) com blocos de 16×16, 8×8 e 4×4, e inter-predição (quadros diferentes) com blocos de 16x16 até 4x4. A transformação e codificação é realizada usando a Transformada Discreta do Cosseno (DCT) que tem os coeficientes quantizados por um valor que é definido pelos fabricantes. O *bitstream* que é codificado e transmitido possui: os coeficientes da transformada quantizados, a estrutura do dado comprimido e as ferramentas usadas, a informação sobre

a sequência completa e outras informações necessárias para fazer as predições (vetor de movimento, resíduos, etc.).

É usado para DVDs de alta definição (*High Definition* - HD DVD), transmissão de televisão de alta definição na Europa, transmissão de televisão para móveis (celular, *smartphones*,*tablet*,etc.), videoconferência, cameras de video comuns.

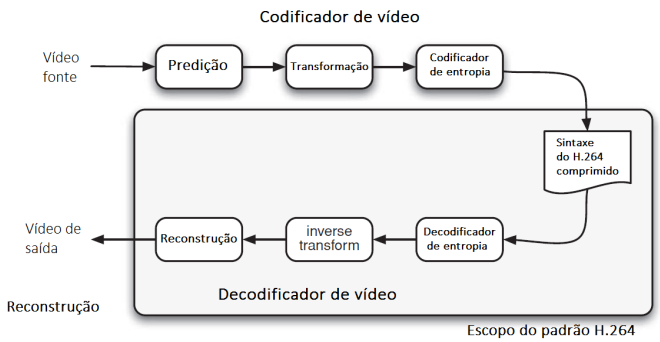


Figura 4.12: Processo de codificar e decodificar do H.264 - adaptado de Richardson [20].

Capítulo 5

Método proposto

Para verificarmos a possibilidade de sequências de DNA serem comprimidas com métodos de compressão com perdas de vídeo, foi usada uma abordagem prática, com o desenvolvimento de um programa que transforma as sequências de DNA do arquivos texto para arquivos de vídeo e depois as transforma novamente só que de vídeo para arquivo texto.

As Seções 5.1, 5.2, 5.3 explicam como ocorre essa transformação a Seção 5.4 explica em detalhes os métodos que fazem essas conversões. Os arquivos de vídeos são reconstruídos a partir do *bitstream* de H.264 e transformados novamente em arquivos de texto.

Esse arquivo de texto reconstruído precisa ser comparado com o arquivo original para verificarmos a precisão na reconstrução. Essa parte não faz parte do sistema do programa, mas nesse trabalho é essencial para analisar os resultados. O método da comparação é descrito na Seção 5.5 assim como as operações matemáticas usadas.

5.1 Processo de codificação e decodificação

A forma como o arquivo de texto é convertido em arquivo de video e vice-versa, para diferenciar do termo codificação de video é chamado de abordagem de vídeo, que no programa atual são:

1. **estável (est)** Os valores são distribuídos entre 8 bits. Detalhes na Subseção 5.4.3
2. **estável 5bits (etv5)** Os valores são distribuídos conforme probabilidade em 5 bits. Detalhes na Subseção 5.4.4
3. **extensão 3 bits (ex3)** Os valores são distribuídos em 3 bits com acesso a arquivo. Detalhes na Subseção 5.4.5;
4. **extensão 4 bits (ex4)** Os valores são distribuídos em 4 bits com acesso a arquivo. Detalhes na Subseção 5.4.6;
5. **extensão 5 bits (ex5)** Os valores são distribuídos em 5 bits com acesso a arquivo. Detalhes na Subseção 5.4.7.

O codificador de vídeo utilizado foi o H.264, implementação x264. O método de compressão usado é com perdas, pois usa um quantizador. Mas o resultado do arquivo texto vai ter que ser reconstruído igualmente.

profile: high

preset: placebo

controle de taxa (ratecontrol): qp (**C**onstant **Q**uantizer) variante com valores de (1, 2, 3, 4, 5, 6, 7, 8, 16, 24, 32, 40)

tune: psnr

alcance da área de busca do estimador de movimento (merange): 128

Como *tune* e *preset* são entradas particulares do x264, na verdade, elas representam tais valores de opções:

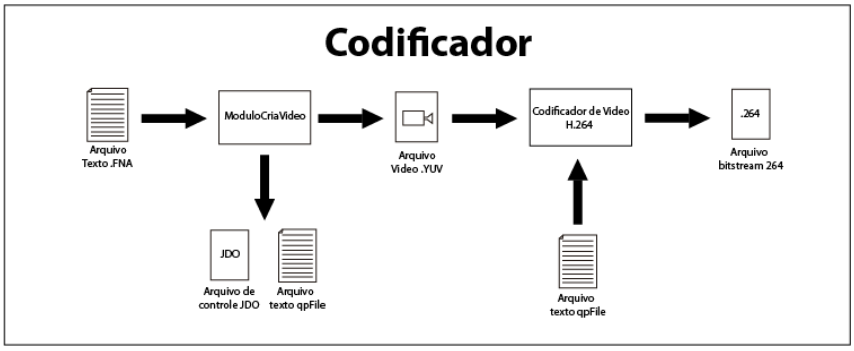
Tune Otimiza o codificador para conseguir bons valores de PSNR, outras entradas da opção são: *Film* para filmes e outros possíveis da vida real, mas não desenhos ou *animes* (Animação Japonesa); *Animation* para conteúdo desenhado; *Grain* para conteúdos velhos ou granulados; *Still Image* para apresentações ou onde tem pouco movimento; *PSNR* para qualidade de *benchmarking*; *SSIM* outro para qualidade de benchmarking; *Fast Devode* otimizado para decodificação rápida que seria utilizado em sistemas com recursos limitados.

Preset Placebo Um conjunto de opções de codificação, é chamada antes dos paramêtros, então as escolhas do usuário prevalecem. As opções são: *me* tesa (procura a estimação de movimento por uma aproximação da Transformada de Hadamard, mais lenta e mais eficiente); *subme* 9 (complexidade de estimação dos subpixels com RDO para todo os quadros); *merange* 24 (alcance da área de busca do estimador de movimento); *ref* 16 (tamanho do buffer das imagens decodificadas - DPB ou Decoded Picture Buffer); *b-adapt* 2 (o algoritmo ótimo, embora lento de escolha de quadros B na predição); *direct* auto (permite que o x264 possa alternar entre predição espacial e temporal quando for necessário); *partitions* all (permite que partições menores que 16x16 sejam usadas, como i8x8, i4x4, p8x8, p4x4, b8x8); *no fast pskip* (desabilita a detecção de saltos "iniciais" na predição dos quadros P); *trellis* 2 (realiza uma quantização de Trellis em todos os modos, ao invés de somente na codificação final de um macrobloco); *bframes* 16 (número máximo de quadros B que o x264 pode usar). Além do *–slow-firstpass*

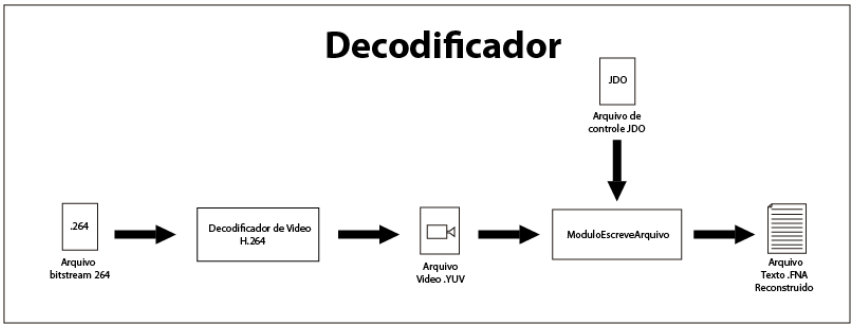
As outras opções estão com o valor *default*. O codificador recebe um arquivo YUV (ou YC_bC_r) 4:2:0, ou seja, as dimensões U e V são metade da largura e altura de Y. O codificador também pode receber um arquivo texto *qpFile* que define como cada quadro vai ser previsto: **I** para predição intra codificado (codificação interna); **P** para predição inter codificado (codificação entre esse quadro e um de referência); **B** para predição inter codificado com dois quadros de referência.

O diagrama na Figura 5.1 representa o fluxo de dados do codec.

Na Figura 5.1(a) é mostrado o caminho feito pelo arquivo na parte do codificador. O arquivo texto passa pelo módulo de criação de vídeo (Seção 5.2) que cria um arquivo de vídeo YUV, um arquivo texto *qpFile*, que será utilizado pelo codificador de vídeo para definir como cada quadro deve ser previsto, e um arquivo texto de controle com o cabeçalho do material genético, o valor da linha e da coluna de cada quadro, um valor de quantos caracteres o arquivo texto tinha por linha e o tipo de abordagem de vídeo. O arquivo de vídeo depois é passado para o codificador H.264 com o arquivo *qpFile* e um valor para o quantizador a ser usado. O quantizador mostra que a compressão usada é



(a) Codificador



(b) Decodificador

Figura 5.1: Diagrama do codec.

com perdas, pois os coeficiêntes do resíduo são modificados. O codificador no final produz um arquivo *bitstream* ".264" que seria a sequência de vídeo comprimida.

Na Figura 5.1(b) é mostrado o caminho feito pelo arquivo na decodificação. O arquivo *bitstream* ".264" passa pelo decodificador H.264 que o transforma novamente em um arquivo de vídeo YUV', que é diferente do arquivo de vídeo do codificador, pois foi quantizado. O arquivo YUV' é passado juntamente com o arquivo de controle para o módulo de escrita do arquivo (Seção 5.3). O módulo escreve arquivo ao mesmo tempo que vai "traduzindo" de arquivo de vídeo para arquivo texto, corrige o erro decorrente da quantização e no final retorna o arquivo texto.

5.2 Módulo Cria Vídeo

Este módulo é responsável por pegar o arquivo texto, transformá-lo em arquivo YUV 4:2:0. Contar os quadros, escrever um arquivo com a predição usada em cada quadro e será usada no codificador H.264, um arquivo de controle com o cabeçalho da sequência genética e outros dados auxiliares. As entradas são: um arquivo ASCII FASTA, a localização onde o vídeo e os arquivos auxiliares serão escritos, a quantidade de linhas, a quantidade de colunas e o tipo de abordagem. O módulo está representado na Figura 5.2 e segue um fluxo sequêncial.

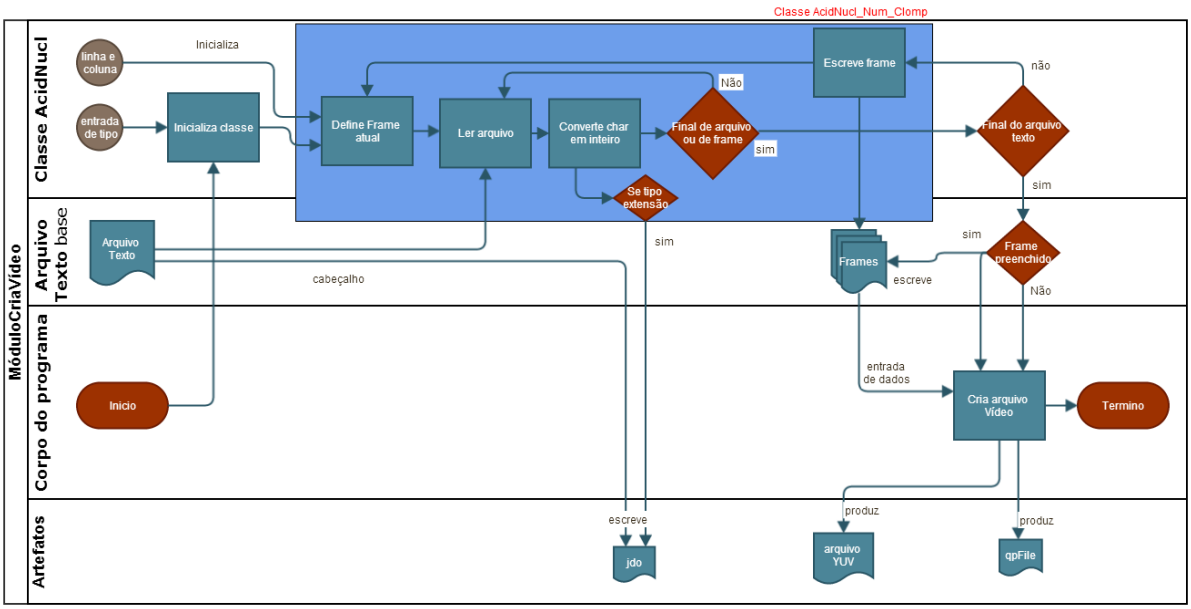


Figura 5.2: Fluxograma do módulo Cria Video.

1. O módulo cria um arquivo de controle que contém o cabeçalho da sequência, e outros dados auxiliares, como a quantidade de linhas e colunas para criar a resolução do quadro e o tipo de abordagem;
2. Enquanto o arquivo não terminar, o módulo lê o arquivo texto, converte os caracteres do arquivo texto para inteiro com a função ac2num e preenche quadros de vídeo;
3. Com os quadros de vídeo produzidos, intercala cada quadro com dois quadros de crominância e grava o vídeo YUV propriamente dito;
4. Além disso cria um arquivo com o tipo de predição para cada quadro que vai ser usado pelo codificador no próximo passo.

5.3 Módulo Escreve Arquivo

Este módulo é responsável por receber o arquivo YUV 4:2:0 e um arquivo de controle e transformá-los em arquivo ASCII FASTA. As entradas são: um arquivo YUV 4:2:0 e o local onde o arquivo vai ser escrito, mas, no presente trabalho, também é essencial o arquivo de controle, não sendo colocado como entrada de dados por ser chamado por contração do nome do arquivo YUV, possibilitando diversos arquivos YUV com quantizadores diferentes da codificação de vídeo usarem um único arquivo de controle, facilitando a análise dos dados. O módulo também é responsável subjetivamente pela correção de erros decorrentes da codificação H.264 O módulo é representado em detalhes na Figura 5.3.

1. Abre o arquivo de controle e obtém os parâmetros auxiliares escritos no arquivo;
2. Percorre os quadros da luminescência, convertendo o valor inteiro dos pixels para caracteres do arquivo texto com a função num2ac, as crominâncias são descartadas;

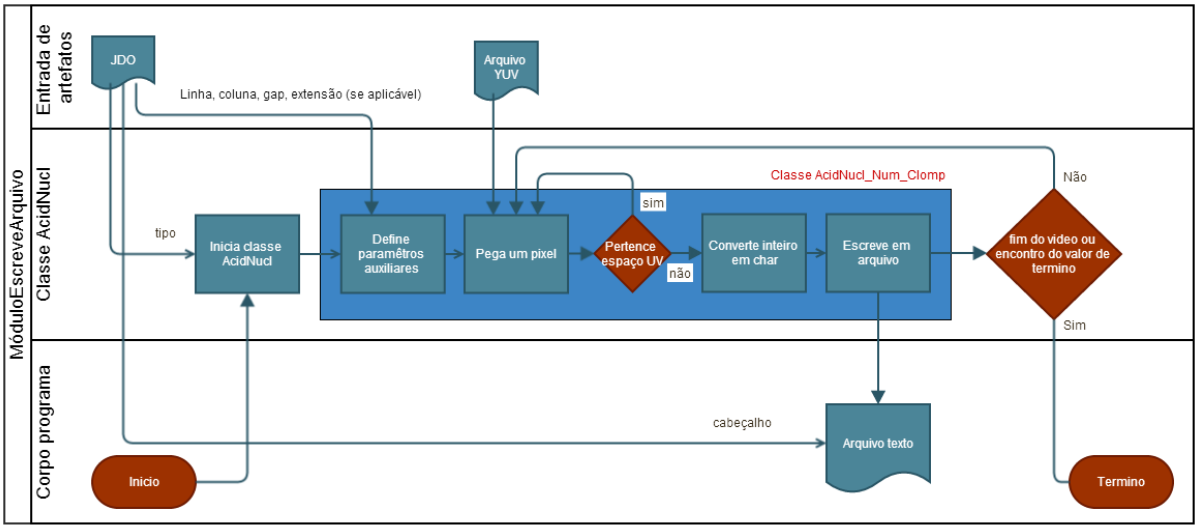


Figura 5.3: Fluxograma do módulo Escreve Arquivo.

3. O erro decorrente da conversão é corrigido na função num2ac.
4. Quando o vídeo termina, o novo arquivo texto também está terminado.

5.4 Classe AcidNucl_Num_Clomp e as abordagens

É o coração do programa, responsável por representar a abordagem de cada tipo para os quadros. Está presente no módulo Cria Vídeo e no Módulo Escreve Arquivo onde trabalha de forma inversa; transformando o caractere de char para um valor inteiro numeral dentro do espaço de imagem e transformando um valor inteiro para um caractere char, respectivamente.

Tendo como base a Tabela 2.3, a classe cria a Tabela 5.1 que possui algumas particularidades como a presença de 'U' representando o uracil do RNA, 'X' representando um código mascarado ou incorreto e de '-' representando uma lacuna de comprimento indeterminado. Essa Tabela representa as bases presentes no DNA (A,C,G,T) e no RNA (A,C,G,U) e as bases ambíguas (R, Y, M, K, S, W, H, B, V, D N, X, -).

A abordagem é o modo como a base é "vista" em termos de caractere e em termos de núméral. Permitindo um conversão entre os dois tipos. Embora cada abordagem tenha os próprios valores, a conversão de caractere para núméral é feito de forma única (um valor, um correspondente), mas a de núméral para caractere é feito por uma faixa de valores (ou *bins*) do núméral para um caractere.

No Código 5.1 são mostrados os atributos da classe:

mtzCharm A matriz com a tabela de caracteres em minúscula;

mtzCharM A matriz com a tabela de caracteres em maiúscula;

mtzNum Vai receber os valores inteiros corespondentes em cada abordagem;

Tabela 5.1: Códigos de bases de ácidos nucléicos.

Símbolo	Base	Significado
G	G	Guanina
A	A	Adenina
T	T	Tinina
U	U	Uracil
C	C	Citosina
R	A ou G	Purina
Y	C ou T	Pirimidina
M	A ou C	Grupo Amina
K	G ou T	Cetona
S	C ou G	Forte interação
W	A ou T	Fraca interação
H	A, C ou T, mas não G	Letra seguinte à G
B	C, G ou T, mas não A	Letra seguinte à A
V	A,C ou G, mas não T(nem U)	Letra seguinte à T (e à U)
D	A, G ou T, mas não C	Letra seguinte à C
N	A, C, G ou T	Qualquer base
X	XXX	Mascarado
-	-	Lacuna de tamanho indefinido

tot A quantidade de entradas que existe na tabela;

difA Valor de subtração para definir o limite inferior ao procurar caractere;

difD Valor de adição para definir o limite supeior ao procurar caractere;

alcanceA Valor de subtração para definir o limite inferior do valor de termino;

alcanceD Valor de adição para definir o limite superior do valor de termino;

vlEXT Valor de extensão;

isExt Booleano para definir se é do tipo extensão;

vlFin Valor de termino quando transforma de imagem para texto, pois é o valor básico do quadro;

pperc Contador interno para percorrer o vetor de caracteres;

Além dos atributos, a classe tem um construtor que inicia com o mnemônico do tipo de abordagem, e dois métodos: ac2num que é executado no Módulo Cria Vídeo e o num2ac que é executado no Módulo Escreve Arquivo.

Código 5.1: Propriedades de classe.

```
1     properties
2         mtzCharm = ['a' 'c' 'g' 't' 'u' 'b' 'd' 'h' 'k' 'm' 'n'
              'r' 's' 'v' 'w' 'x' 'y' '-'];
3         mtzCharM = ['A' 'C' 'G' 'T' 'U' 'B' 'D' 'H' 'K' 'M' 'N'
              'R' 'S' 'V' 'W' 'X' 'Y' '-'];
4         mtzNum;
5         tot = 18;
6         difA;
7         difD;
8         alcanceA;
9         alcanceD;
10
11        vlEXT;
12        isExt;
13        vlFIN;
14        pperc;
15    end
```

5.4.1 Método ac2num

O **ac2num** está ilustrado na Figura 5.4. Executada no módulo de cria vídeo, converte o caractere recebido em um valor de inteiro compatível com a escala monocromática (8 *bits*). A implementação pode aceitar uma entrada de arquivo para escrever quando necessário (Subseções 5.4.5, 5.4.6, 5.4.7).

1. O caractere, enquanto não for encontrado, é comparado com os outros da tabela;
2. Se o arquivo for DNA, um valor deve ser encontrado; senão retorna 0;
3. Se a abordagem for do tipo extensão, verifica se o caractere pertence ao grupo de extensão; negativo em qualquer dos casos, retorna o valor inteiro correspondente à tabela;
4. A abordagem sendo extensão e o caractere pertencente ao grupo de extensão, o caractere é gravado no arquivo de controle por meio do ponteiro, e o valor inteiro de extensão é retornado.

5.4.2 Método num2ac

O **num2ac** está ilustrado na Figura 5.5. Executada no módulo de escreve arquivo, converte o valor inteiro, de monocromática (8 *bits*), que deve estar em uma das faixas de valores (dentro dos *bins* de um valor) para um caractere. Essa faixa de valores funciona como uma correção de erros, pois na quantização é esperado que o valor não seja o mesmo, mas próximo do valor que era. Porém pode retornar 0 caso fique em uma região nula (Subseções 5.4.4, 5.4.5, 5.4.6, 5.4.7) A implementação pode aceitar uma entrada de arquivo para ler quando necessário (Sebbseções 5.4.5, 5.4.6, 5.4.7).

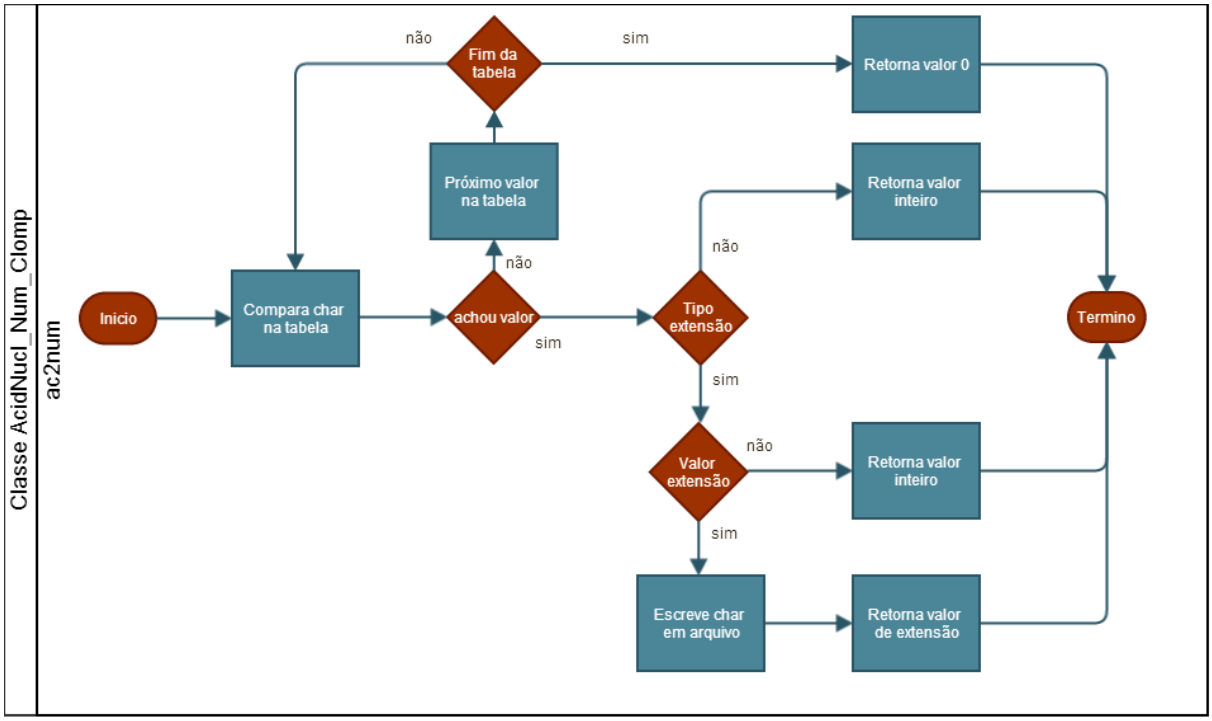


Figura 5.4: Fluxograma da função ac2num.

1. Primeiro verifica se o valor inteiro pertence a faixa de valores (*bins*) do valor de termino, pertencendo retorna um caractere de termino, se não continua o processo;
2. Depois procura nas faixas de valores existentes, qual o valor de entrada pertenceria.
3. Se não encontrar uma faixa, retorna 0.
4. Se a abordagem for do tipo extensão, e o valor pertencer a faixa de valor de extensão, pega um caractere no vetor auxiliar e o retira dele, se o vetor estiver vazio retorna um valor *default* de erro de caractere de extensão.
5. Se não for do tipo extensão ou do valor de extensão, se for do tipo extensão, retorna o caractere achado.

5.4.3 Abordagem Estável

Mnemônico: est

Primeira abordagem a ser desenvolvida, Abordagem Estável (EST) foi pensada como uma transformação simples para sedimentar os métodos e práticas. Essa abordagem tem uma possibilidade para trabalhar com RNA sem grandes prejuízos, pois a base U tem o próprio valor separado. Também considerada 8 *bits* porque os valores, de modo geral, fazem uso de todos os *bits* da imagem, ao dividir as entradas igualmente entre as 255 intensidades de uma imagem monocromática. Forçando o codificador "H.264" a interpretar o mesmo. A distância entre cada valor foi de 14 unidades de intensidade, a

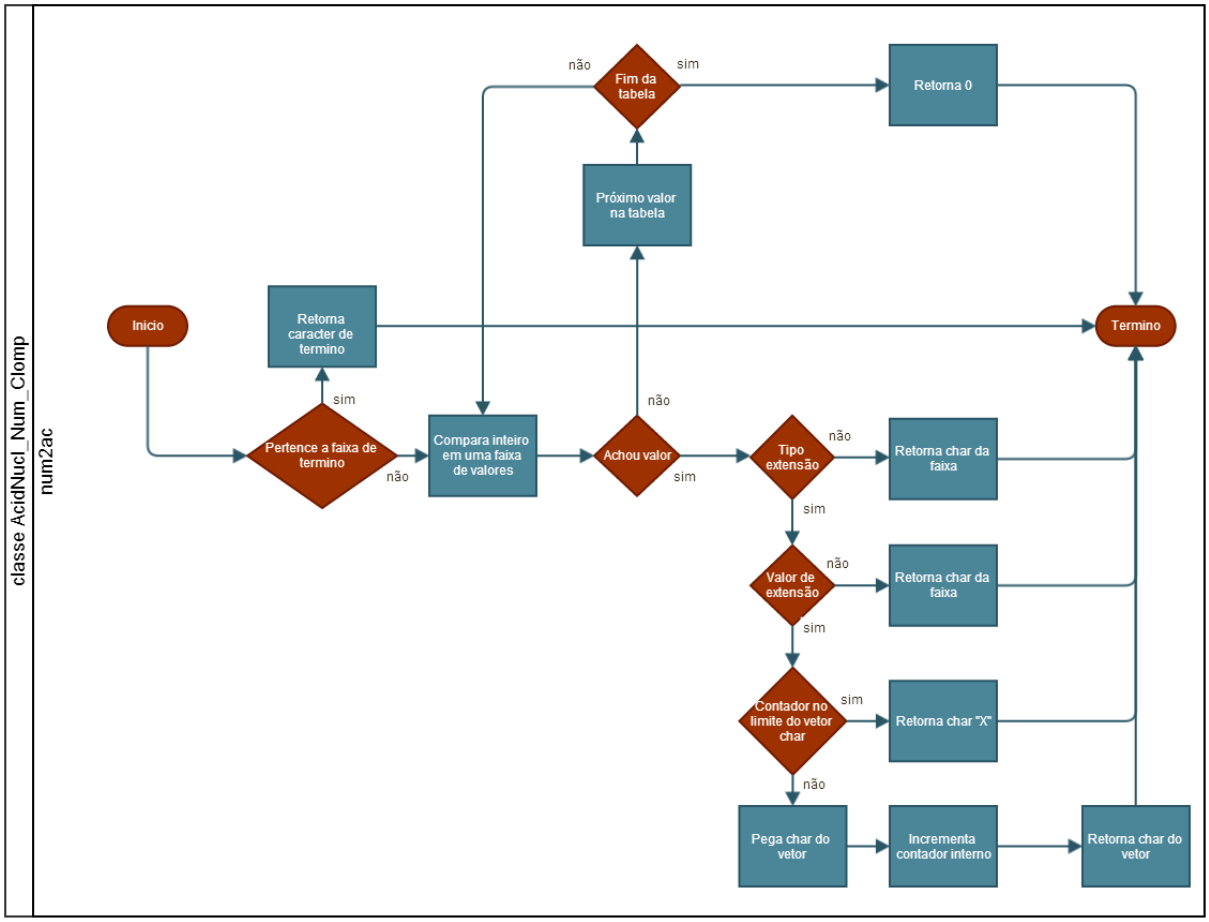


Figura 5.5: Fluxograma da função num2ac.

área do valor então é considerada 7 valores para cima e 7 valores para baixo resultando na Tabela 5.2, qualquer intensidade que se encaixe nessa faixa é considerado desse valor. Privilegia a estabilidade e precisão na conversão do arquivo em detrimento da compressão.

5.4.4 Abordagem Estável 5 bits

Mnemônico: etv5

Abordagem Estável de 5 *bits* (ETV5) foi pensada em simular uma redução de *bits* enquanto aproveita a probabilidade de alguns valores serem mais abundantes que outros. Essencialmente como é uma sequência de DNA, as que tem maior probabilidade são as bases principais do DNA (A, C, G, T) elas recebem um distanciamento maior entre outros valores do que as bases ambíguas (H, W, D, R, B, X, Y, M, V, S, K, N e -). Como 5 *bits* tem 32 possibilidades e temos 18 entradas, para as bases principais terem uma folga maior entre os valores, muitas das outras bases vão ter poucos valores extras para corrigir erro. Isso resultou em um espaçamento de 2 intensidades (para cima e para baixo) para as bases principais e a base N, gerando 4 valores de correção ao total, e de 1 intensidade de espaçamento para a base Y com 2 valores de correção ao total. As outras bases não tiveram valores de "escape" ficando restritas ao único valor. Para ter mais folga, a base U

Tabela 5.2: Tabela abordagem estável.

mtzChar	valor mtzNum	difA/alcanceA	difD/alcanceD	limite inferior (inclusivo)	limite superior (exclusivo)
vlFIN	0	7	7	-7	6
A	252	7	7	245	259
T	182	7	7	175	189
G	112	7	7	105	119
C	42	7	7	35	49
U	196	7	7	189	203
H	238	7	7	231	245
W	224	7	7	217	231
D	210	7	7	203	217
R	168	7	7	161	175
B	154	7	7	147	161
X	140	7	7	133	147
Y	126	7	7	119	133
M	98	7	7	91	105
V	84	7	7	77	91
S	70	7	7	63	77
K	56	7	7	49	63
N	28	7	7	21	35
-	14	7	7	7	21

Tabela 5.3: Tabela Abordagem estável 5 bits.

mtzChar	valor mtzNum	difA/alcanceA	difD/alcanceD	limite inferior (inclusivo)	limite superior (exclusivo)
vlFIN	128	32	32	96	160
A	31	2	2	29	33
T	27	2	2	25	29
G	19	2	2	17	21
C	23	2	2	21	25
N	15	2	2	13	17
Y	12	1	1	11	13
-	10	0	1	10	11
X	9	0	1	9	10
W	8	0	1	8	9
V	7	0	1	7	8
S	6	0	1	6	7
R	5	0	1	5	6
M	4	0	1	4	5
K	3	0	1	3	4
H	2	0	1	2	3
D	1	0	1	1	2
B	0	0	1	0	1
U	0	0	1	0	1

do que poderia ser usado no RNA, vai ter o valor de outra entrada ambígua , tornando essa abordagem exclusiva do DNA. O resultado de cada faixa de valor pode ser conferido na Tabela 5.3.

Pela Tabela também notamos que existe uma área que não tem nenhum valor (da intensidade 33 a intensidade 95 e da intensidade 160 a 255), as chamadas áreas nulas, que representam quanto o quantizador influiu nas intensidades do quadro.

5.4.5 Abordagem Extensão 3 bits

Mnemônico: ex3

A Abordagem Extensão de 3 *bits* (EX3) foi desenvolvida para maximizar a redução de *bits* enquanto ainda poderia reconstruir o arquivo original. Aprimorando a ideia do ETV5 de conferir mais faixas de escape para os valores mais prováveis, as bases principais (A, C, G, T) são certezas absolutas de estarem no arquivo, mas as outras entradas por serem ambíguas não. Se essas bases estiverem vão ser em número bem menor que as bases principais, para diferenciar serão chamadas de bases de extensão (H, W, D, R, B, X, Y, M, V, S, K, N, - e U). Assim surgiu a ideia de transformar as outras entradas em uma única, aumentando a redução de *bits* e usar um arquivo auxiliar para ajudar a manter um rastro da sequência de "caracteres extras"do DNA.

Tabela 5.4: Tabela abordagem extensão 3 bits.

mtzChar	valor mtzNum	difA/alcanceA	difD/alcanceD	limite inferior (inclusivo)	limite superior (exclusivo)
vlFIN	128	64	64	64	192
A	0	0	2	0	2
T	4	0	2	4	6
G	2	0	2	2	4
C	6	0	1	6	7
N, Y, W, X, V, K, M, R, S, -, U, B, D, H	7	0	1	7	8

Como o arquivo de controle já guarda o cabeçalho da sequência, linha, coluna, quantidade de caracteres por linhas de DNA e o tipo da abordagem, poderia também guardar os caracteres de extensão. Com apenas 5 entradas das 18 iniciais a quantida mínima de *bits* que poderia comportar essas entradas é 3 *bits*. Essa abordagem pode ser usada para representar RNA, mas não é proveitoso, já que o U ficaria no lugar do T, e o U como um valor de extensão, tornaria o arquivo de controle uma separação de um dos componentes da imagem sem o processamento do codificador de vídeo.

A redução é máxima com 3 *bits*, mas a distância entre as entrada fica prejudicada, pois só possibilita 2 intensidades entre cada valor. Foi escolhido, sem qualquer razão, que as bases principais ficassem com 2 intensidades para cima, mas um delas teria que ficar com um único valor (sem intensidades auxiliares para compensar o erro de quantização), assim como o valor que representaria as bases de extensão também teria somente um valor (consequentemente sem intensidades auxiliares para compensar o erro do quantizador).

Isso resultou na Tabela 5.4 cuja área nula é da intensidade 8 até 63 e da 192 até 255, essa área não é codificada, mas ajuda a entender quando a correção (do método num2ac) é insuficiente pela mudança causada pelo quantizador.

5.4.6 Abordagem Extensão 4 bits

Mnemônico: ex4

A Abordagem Extensão de 4 *bits* (EX4) foi desenvolvida para melhorar a precisão enquanto ainda mantém um número reduzido de *bits*. Com EX3 a redução foi máxima dentro das condições, mas o espaço de erro (*bins* das intensidades de correção) também foi pouco o que possibilita muitos erros. Como uma alternativa no meio termo pode ter resultados melhores, aumentando de 3 *bits* para 4 *bits* dobramos as possibilidades, obtendo mais espaço para erro (mais *bins* entre os valores) o que deve permitir uma precisão maior na decodificação do arquivo. Tanto as bases principais (A, C, G, T) como as de extensão (H, W, D, R, B, X, Y, M, V, S, K, N, - e U) puderam ter a mesma distância, de 1 intensidade (no geral, pois o limite superior é exclusivo), mostrando mais

Tabela 5.5: Tabela abordagem extensão 4 bits.

mtzChar	valor mtzNum	difA/alcanceA	difD/alcanceD	limite inferior (inclusivo)	limite superior (exclusivo)
vlFIN	128	64	64	64	192
A	2	1	2	1	4
T	5	1	2	4	7
G	8	1	2	7	10
C	11	1	2	10	13
N, Y, W, X, V, K, M, R, S, -, U, B, D, H	14	1	2	13	16

equilíbrio na conversão e correção de erro que EX3 resultando em faixas de 3 *bins* de correção. Resultando na Tabela 5.5 cuja área nula é da intensidade 16 até 63 e da 192 até a 255. Por pertencer a mesma categoria do EX3, sofre do mesmo mal em relação ao RNA, pode representar, mas não irá se beneficiar, nem ser melhor do que o DNA.

5.4.7 Abordagem Extensão 5 bits

Mnemônico: ex5

A Abordagem Extensão de 5 *bits* (EX5) foi desenvolvido para analisar a eficiência do tipo extensão comparando-o com o ETV5 já que possuem a mesma simulação de redução de *bits*. Pois EX3 e EX4 pareceram promissores e fica a dúvida de quanto benefício acarretará para a compressão e a precisão o aumento do número de *bits*. Devido a ter mais intensidades livres do que ETV5, as bases principais (A, C, G, T) e a de extensão (H, W, D, R, B, X, Y, M, V, S, K, N, - e U) tiveram 7 *bins* de correção. Resultando na Tabela 5.6 cuja área nula é de 31 até 95 e de 160 até 255.

5.5 Processos de análise

Em vídeo e imagem, o produto reconstruído não precisa ser necessáriamente igual, pois dentro de uma faixa de intensidades próximas o olho humano não nota muita diferença. Mas com textos a reconstrução tem que ser igual porque cada informação é importante. E com esse trabalho não poderia ser diferente, pois embora estamos passando pelo domínio da imagem o resultado final é um arquivo texto, e mais importante uma sequência de DNA, uma mudança de caractere pode mudar, se for na região de codificação de proteína, pode mudar uma proteína essencial e a sequiencia não terá mais valor, pois não é mais confiável.

Por isso foi desenvolvida uma função que embora não faça parte do ciclo de codificação e decodificação é essencial para a parte da análise, pois fornece dados para criar um gráfico e facilitar o entendimento do programa.

Tabela 5.6: Tabela abordagem extensão 5 bits.

mtzChar	valor mtzNum	difA/alcanceA	difD/alcanceD	limite inferior (inclusivo)	limite superior (exclusivo)
vlFIN	128	32	32	96	160
A	4	4	3	0	7
T	22	3	3	19	25
G	16	3	3	13	19
C	10	3	3	7	13
N, Y, W, X, V, K, M, R, S, -, U, B, D, H	28	3	3	25	31

A função recebe como entrada: o arquivo original e o arquivo reconstruído. Retorna a porcentagem de acerto do arquivo reconstruído, a quantidade de caracteres do original e a quantidade de caracteres acertados do arquivo reconstruído. Possibilitando verificar a precisão da conversão. Internamente a função possui um contador de caracteres individual para o original, para o reconstruído para os caracteres iguais uma tabela igual a Tabela 5.1 com uma dimensão acrescida (desconhecido) e uma matriz quadrada da Tabela 5.1 com uma dimensão acrescida para noção de comparação, com as linhas pertencendo ao arquivo original e as colunas ao arquivo reconstruído.

Essa função executa tal sequência de passos:

1. Abre o arquivo original e conta a quantidade de cada base que existe no arquivo;
2. Abre o arquivo reconstruído e compara linha por linha com o arquivo original.
3. Marca a quantidade de linhas comparadas, de caracteres iguais nos dois arquivos, quantos caracteres a mais foram encontrados no arquivo original e quantos caracteres a mais foram encontrados no arquivo reconstruído e a quantidade de vezes que uma base foi trocada por outra.
4. Mostra quantos caracteres foram iguais a porcentagem;
5. Mostra se for o caso, quantos caracteres faltaram serem reconstruídos e a porcentagem.
6. Mostra, para cada base, a quantidade de caracteres trocados por outro e a porcentagem.
7. Escreve em arquivo, caso exista, caracteres faltantes.

Desejável para esse programa um acerto de 100%. Embora a porcentagem mostrado é relativa, pois a quantidade de dados pode ser tão grande que pequenos erros não aparecem na probabilidade, sendo necessário nesses casos uma análise discreta.

Porém somente analisar o arquivo texto que é a última etapa, seria uma falha ao não ver todo o potencial do programa. O programa cria vários artefatos que estão interligados, os arquivos YUV, os arquivos de texto FASTA e principalmente o *bitstream* 264 que é a espinha dorsal na qual o trabalho se apoia. O arquivo FASTA é convertido em vídeo para poder virar um *bitstream* 264. O codificador de vídeo cria o *bitstream* e o decodificador o traduz.

Uma boa forma de analisar o H.264 é pela taxa de *bits*. Para melhor analisarmos a diferença entre os *bitstream*, é necessário ver a semelhança entre o arquivo YUV e o *bitstream*.

$$VidYUV \xrightarrow{codificador} H264 \xrightarrow{decodificador} VidYUV' \tag{5.1}$$

A taxa total de *bits* por símbolo de um arquivo de vídeo pode ser medida pela seguinte equação

$$BitsPorSimbolo = 8 \cdot \frac{tamArquivoYUV}{numeroFrames \cdot numLinhas \cdot numColunas} \tag{5.2}$$

Embora H.264 seja um *bitstream* podemos supor que assemelhe a um arquivo YUV. Por definição um arquivo YUV 4:2:0 tem uma taxa total de *bits* de 12. Porque a crominância (U e V) aparece entre cada 4 bits de luminância (Y). E como a crominância ocorre pela camada U e V são 2 a cada 4 bits. Um píxel de Y possui 8 *bits* e 4 pela crominância resultando em 12. Porém poderiamos "descartar" a contagem da crominância com a seguinte equação

$$BitsPorSimbolo = 8 \cdot \frac{tamArquivoYUV}{(numFrames \cdot numLin \cdot numCol) + \left(numFrames \cdot \frac{numLin}{2} \cdot \frac{numCol}{2}\right)} \tag{5.3}$$

Sem prejuízo de fato pois só mudamos o espaço do contradomínio. E como $VidYUV$ e $VidYUV'$ tem a mesma taxa com a Equação 5.3 ela pode ser aplicada em H.264, para facilitar a amostragem.

A Equação 5.2 é melhor para vídeos YUV 4:0:0, enquanto a Equação 5.3 é mais indicada para vídeos YUV 4:2:0.

Mas é necessário verificarmos a qualidade da compressão. Isso pode ser obtido calculando o PSNR da Equação 3.13 entre o arquivo YUV criado antes da codificação e o arquivo YUV' criado depois da decodificação. Relembrando que valores maiores de PSNR são melhores pois o erro quadrático médio foi menor.

Como a qualidade do arquivo YUV e do arquivo texto passam pelo H.264, os gráficos de porcentagem de acertos e PSNR foram feitos em relação a taxa total do H.264.

Também foi analisado a taxa de compressão definida pela Equação 3.2. Onde valores maiores são melhores, mas devido a natureza do problema é limitado pela porcentagem de acerto e se houve alguma troca.

Capítulo 6

Resultados experimentais

Foram usadas os formatos FASTA das pastas CHR_I, CHR_II e CHR_IV do banco de dados GenBank do NCBI em [8]. Para facilitar a escrita as abordagens vão ser escritar como mnemônicos.

6.1 Sequência 1

Arquivo: NC_003070

Tamanho Mb: 28,6

Tamanho Bytes: 30.063.834

Quantidade de caracteres: 29.640.317

Tamanho RAR bytes: 8738510

Taxa compressão do RAR: 3,4403

Entropia do texto: bits 1,9864

Primeiro foi executado a abordagem EST, cujos PSNR e porcentagem de acerto estão do lado esquerdo da Figuta 6.1. A EST forneceu valores muito bons de PSNR e precisão exceto nos quantizadores 32 e 40, que, inclusive, não tiveram uma boa reconstrução de arquivo. Porém deixou muito a desejar em questões de tamanho (Tabela 6.1) de arquivo e taxa de compressão (Tabela 6.2).

A abordagem ETV5, lado direito da Figura 6.1, não teve um PSNR inicial tão elevado quanto o EST, mas o PSNR final foi maior, mostrando que o ETV5 não tem uma variação tão brusca nos valores. Obteve uma porcentagem de acertos regular, mas aceitável, no final a queda foi muito abrupta, mas ainda ficou acima de 30%. Em relação ao tamanho (Tabela 6.1), melhorou muito comparado ao EST, no quantiador 32 e 40, porém a queda foi mais surpreendente. A taxa de compressão (Tabela 6.2) em relação a EST desde o inicio foi bem melhor, nos três últimos quantizadores (24, 32, 40) o salto foi grande, mas a precisão também caiu bem.

A abordagem EX3, no lado esquerdo da Figura 6.2, foi a melhor em termos de tamanho (Tabela 6.1) até quantizador 32 e quantizador 40, onde ETV5 deu uma queda abrupta;

mas em precisão, foi a pior. Talvez a questão do tamanho deva-se em relação ao arquivo de controle, mas a abordagem extensão em termos gerais se comportou mais "equilibrada" que as abordagens de tipo estável, pois o tamanho se comportou de forma previsível. A taxa de compressão (Tabela 6.2) de EX3 realmente foi a melhor, como o imaginado, mas para isso teve o custo da precisão, já que nem no quantizador 1 a abordagem teve 100% de reconstrução.

A abordagem EX4, no lado direito da Figura 6.2, teve um percentual de acerto melhor que EX3, embora o PSNR não tenha sido melhor, possivelmente devido as faixas de correção do EX3 serem irregulares O tamanho (Tabela 6.1), em relação a EX3, aumentou relativamente pouco assim como a taxa de compressão (Tabela 6.2). Foi a segunda melhor no geral, mas era previsto já que também é o segundo com menor número de *bits*. A melhora da precisão na reconstrução surpreendeu positivamente.

A abordagem EX5, do lado esquerdo da Figura 6.3, teve um leve aumento de precisão, mas uma queda acentuada perto do final. O PSNR do EX5 surpreendentemente foi pior que ETV5, que mesmo com uma faixa de intensidades menor foi melhor que EX5. A previsão inicial da abordagem seria que estivesse relativamente próxima ou melhor que ETV5, mas EX5 se mostrou abaixo das expectativas, tanto pelo PSNR, quanto pelo tamanho (Tabela 6.1) e pela taxa de compressão (Tabela 6.2), por um leve aumento da precisão inicial.

No lado direito da Figura 6.3, todos os gráficos foram plotados juntos para uma comparação mais efetiva. O percentual de acerto das abordagens ficaram muito próximas, o PSNR ficou bem espaçado, com ETV5 quase tocando EX4. Na Tabela 6.2 as células com cor azul são as que foram verdadeiramente 100%, sem nenhuma troca. Indiretamente isso coloca um limite da taxa de compressão verdadeiramente sem perdas de 1,78 obtida por EX4 nos valores inicias, e exemplifica o verdadeiro problema: a reconstrução tem que ser sem erros, mas com uma grande quantidade de dados, o percentual "mascarará" os erros.

Como a utilidade de um algoritmo de compressão é medida pela taxa de compressão. As taxas foram comparadas com um compactador padrão $WinRAR^{®}$. E infelizmente os valores que mais se aproximam de 3,44 pertencem a faixa do quantizador 16, que já apresentam erros entre 10% a 45%.

6.2 Sequência 2

Arquivo: NC_003071

Tamanho Mb: 19,0

Tamanho Bytes: 19.924.328

Quantidade de caracteres: 19.643.621

Tamanho RAR bytes: 5.845.279

Taxa compressão do RAR: 3,4086

Entropia do texto: bits 1,9417

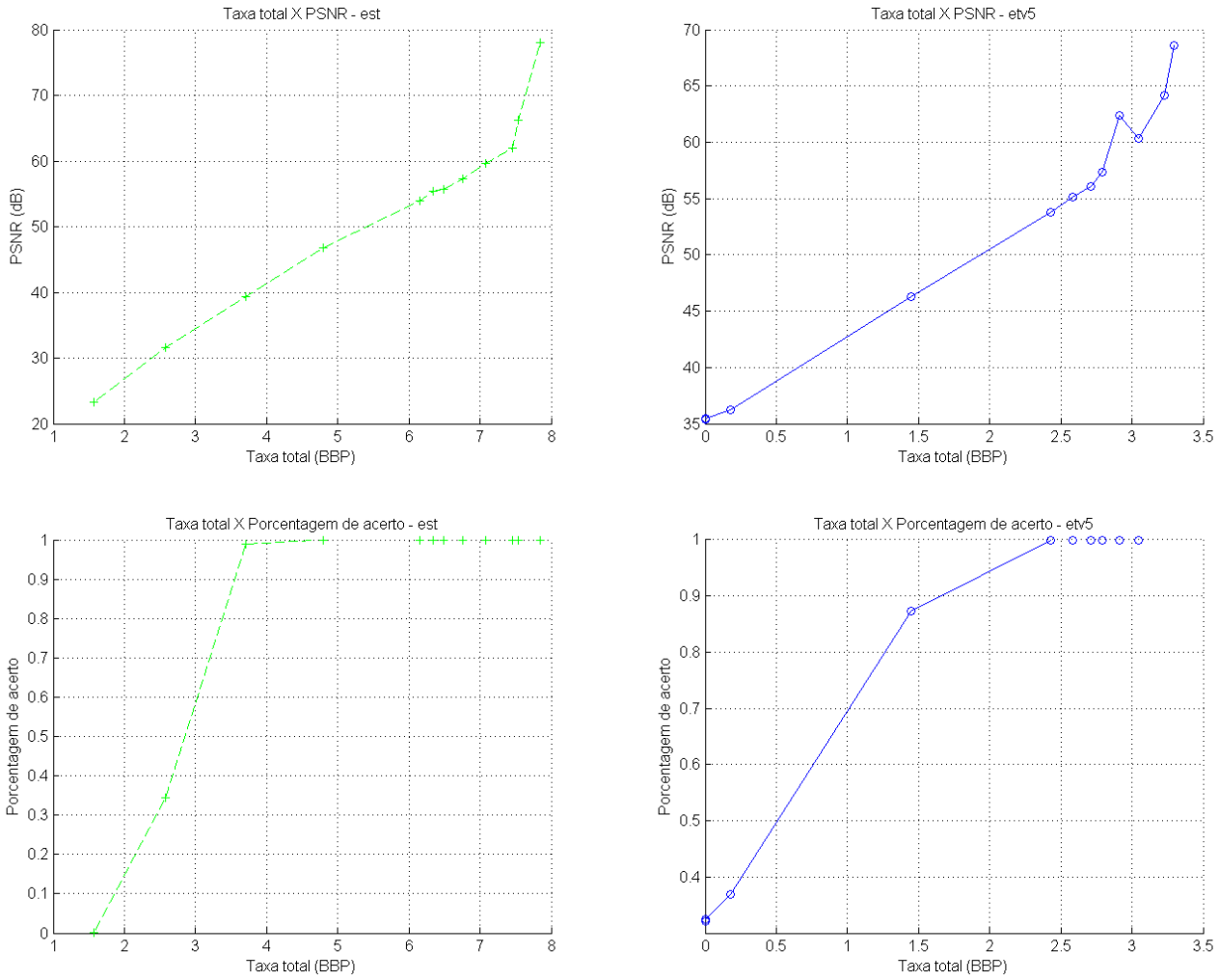


Figura 6.1: Gráficos das abordagens Estável e Estável 5 bits para NC003070.

Tabela 6.1: Tamanho do H264 junto com arquivo de controle para NC003070.

qp	est	etv5	ex3	ex4	ex5
1	43.971.282	18.445.156	14.899.332	16.917.914	22.810.015
2	42.205.330	18.074.338	14.308.745	16.528.923	22.049.126
3	41.740.054	17.054.644	13.704.308	16.056.652	21.394.589
4	39.653.276	16.327.661	13.108.278	15.089.426	20.855.993
5	37.886.733	15.649.139	12.235.077	14.183.611	19.985.520
6	36.354.034	15.198.343	11.526.269	13.665.653	19.085.904
7	35.495.301	14.480.059	10.884.881	13.015.654	18.597.217
8	34.481.568	13.606.389	10.071.122	12.178.216	17.594.909
16	26.850.621	8.124.236	4.565.820	6.692.742	11.705.278
24	20.777.871	1.000.148	218.643	268.081	6.079.331
32	14.474.299	8.486	209.126	214.175	241.500
40	8.767.794	5.140	202.227	203.501	203.585

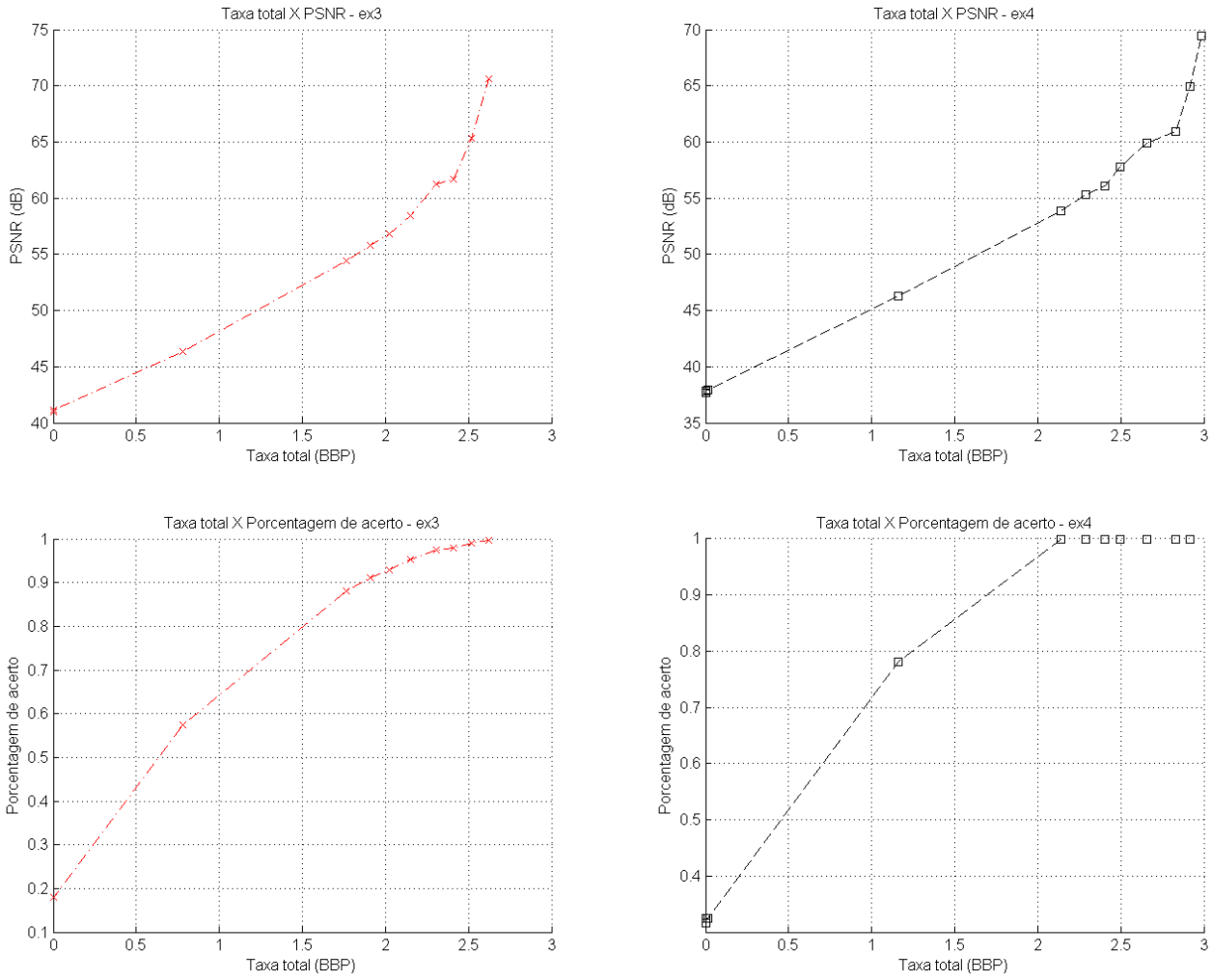


Figura 6.2: Gráficos das abordagens Extensão 3 bits e Extensão 4 bits para NC003070.

Tabela 6.2: Taxa de compressão para o FASTA NC003070.

Abord.	qp 1	qp 2	qp 3	qp 4	qp 5	qp 6	qp 7	qp 8	qp 16	qp 24	qp 32	qp 40
est	0,68	0,71	0,72	0,76	0,79	0,83	0,85	0,87	1,12	1,45	2,08	3,43
etv5	1,63	1,66	1,76	1,84	1,92	1,98	2,08	2,21	3,70	30,06	3542,76	5848,99
ex3	2,02	2,10	2,19	2,29	2,46	2,61	2,76	2,99	6,58	137,50	143,76	148,66
ex4	1,78	1,82	1,87	1,99	2,12	2,20	2,31	2,47	4,49	112,14	140,37	147,73
ex5	1,32	1,36	1,41	1,44	1,50	1,58	1,62	1,71	2,57	4,95	124,49	147,67

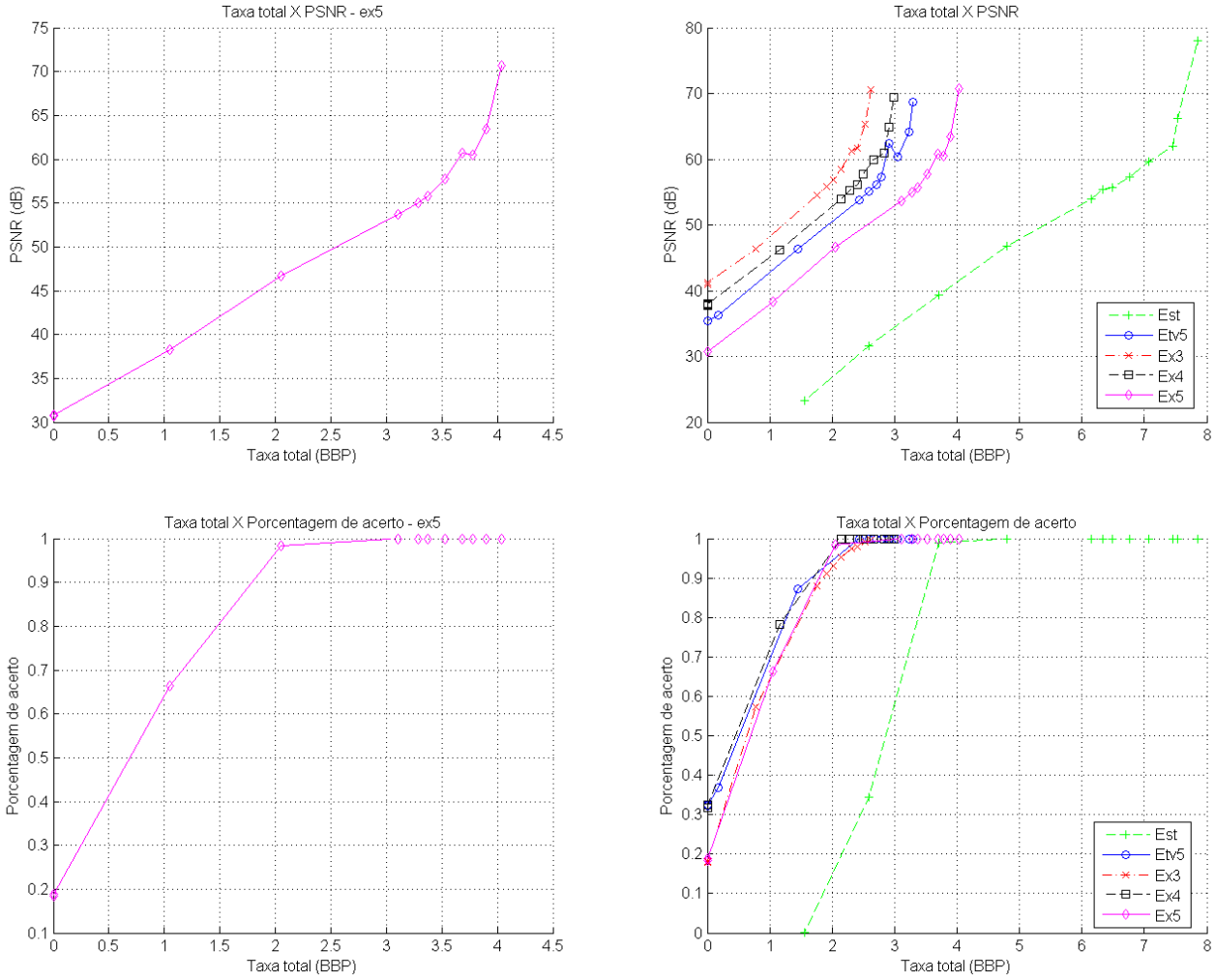


Figura 6.3: Gráficos das abordagens Extensão 5 bits e Todas as abordagens juntas para NC003070.

Muitas das considerações feitas anteriormente se encaixam para a 2ª sequência. Mostrando que o programa está proporcional para o grande maioria das abordagens, na Figura 6.6 do lado direito só a porcentagem de acerto dos últimos quantizadores da abordagem EST melhoraram um pouco.

A abordagem EST (lado esquerdo da Figura 6.4) continuou com os maiores tamanhos (Tabela 6.3) e precisão sobre aumento de quantizador, mas também as menores taxas de compressão (Tabela 6.4). Curiosamente no quantizador 32 teve uma melhor reconstrução que NC_003070, no quantizador 40 em ambas as sequências a reconstrução foi pífia (0,01% e 0,00%).

A abordagem ETV5 (lado direita da Figura 6.4) teve um comportamento esperado. Era previsto que erraria mais em comparação com NC_003070, em termos da Tabela 6.4, pois a distribuição das entradas de incerteza, embora tentando ser o mais imparcial possível, segue um pouco o estilo, das entradas presentes no arquivo NC_003070, que tem algumas bases ambíguas que não tem nesse arquivo, resultando em mais trocas nos quantizadores iniciais.

A abordagem EX3 (lado esquerdo da Figura 6.5) seguiu como o que produz os menores tamanhos (Tabela 6.3) e melhores taxas de compressão (Tabela 6.4), exceto no quantizador 32 onde perdeu para o ETV5, e ficou ligeiramente melhor no quantizador 40.

A abordagem EX4 (lado direito da Figura 6.5) e a EX5 (lado esquerdo da Figura 6.6) tiveram, proporcionalmente, o mesmo desempenho que na sequência NC_003070.

Quando todas as abordagens são colocadas juntas (lado direito da Figura 6.6), comparando com a Figura 6.3 nota-se que as abordagens ETV5 e EX4 em termos de precisão são praticamente iguais, com uma leve vantagem para ETV5, depois a EX3 e EX5 e por último a EST, que melhorou um pouco a precisão para os quantizadores mais altos. O PSNR é mais distinguível entre as abordagens, sendo a ordem da melhor para a pior, EX3, EX4, ETV5, EX5 e EST, quando busca um equilibrio entre taxa baixa e PSNR alto. Comparando as taxas de compressão das verdadeiras reconstruções perfeitas (Tabela 6.4), o EX4 novamente é o melhor com 1,79. Levemente melhor que com NC_003070 (Tabela 6.2), mas ainda assim baixo se compararmos com o $WinRAR^{®}$. Os resultados que podem competir com o $WinRAR^{®}$ são do quantizador de valor 16 em diante, para ETV5, EX3 e EX4, do quantizador 24 em diante para EX5 e acima de 40 para EST, mas a faixa de erro está entre 10% a 45%. O que para um arquivo texto com dados sensíveis, muito alto. Nota-se também o salto que ocorre,de inicio e mais lento, no quantizador 16, mas intensifica-se do quantizador 32 em diante na redução de tamanho dos arquivos, no quantizador 40 inclusive ocorre leves erros de reconstrução.

6.3 Sequência 3

Arquivo: NC_003073

Tamanho Mb: 16,9

Tamanho Bytes: 17.800.319

Quantidade de caracteres: 17.549.528

Tamanho RAR bytes: 5.189.289

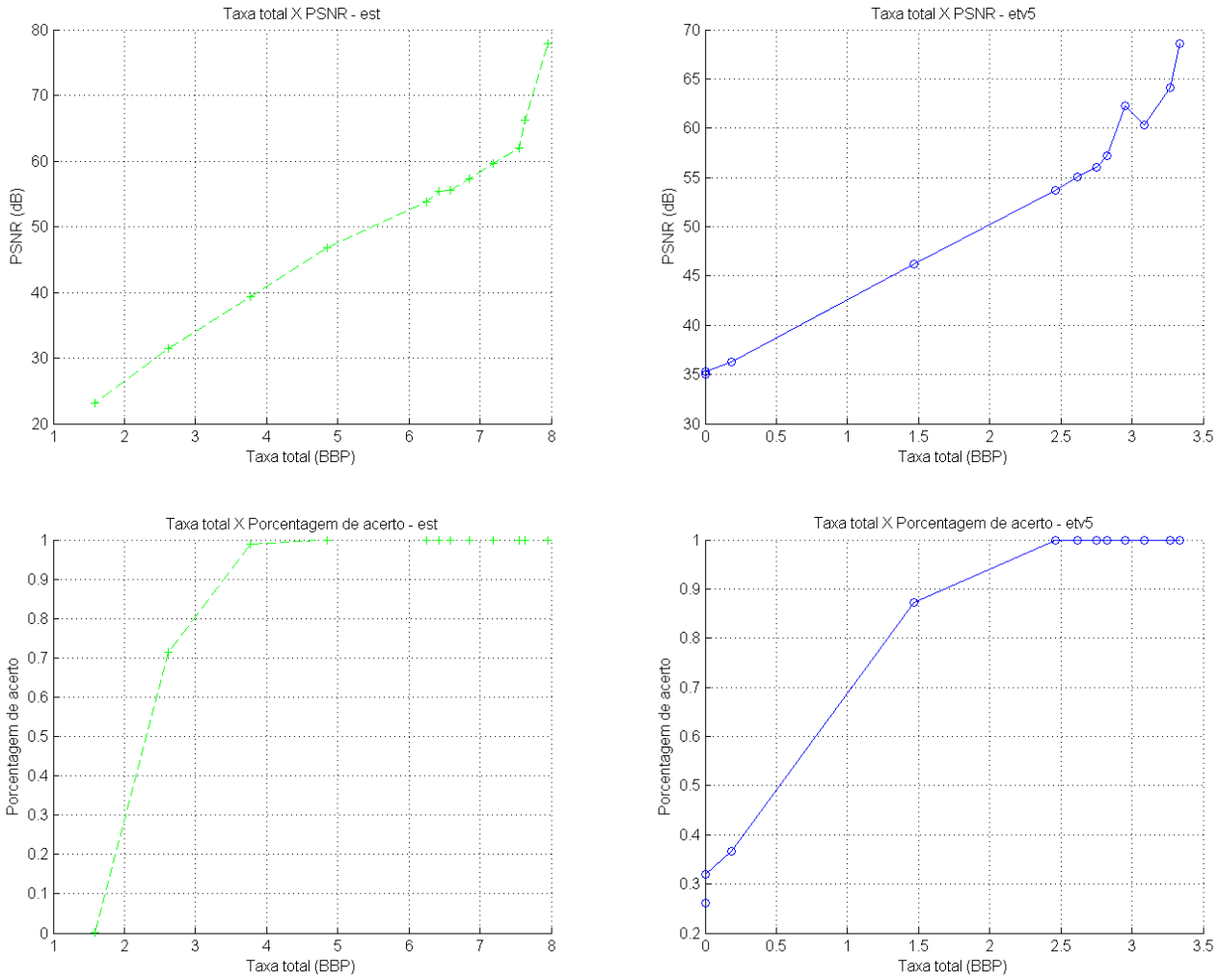


Figura 6.4: Gráficos das abordagens Estável e Estável 5 bits para NC003071.

Tabela 6.3: Tamanho do H264 junto com arquivo de controle para NC003071.

qp	est	etv5	ex3	ex4	ex5
1	29.313.027	12.298.066	9.808.667	11.151.994	15.075.909
2	28.143.480	12.049.132	9.417.192	10.884.540	14.575.544
3	27.847.831	11.376.960	9.013.009	10.585.494	14.143.046
4	26.474.319	10.878.427	8.600.752	9.949.001	13.789.244
5	25.258.664	10.410.751	8.054.251	9.355.052	13.199.890
6	24.262.438	10145567	7.561.965	8.998.400	12.608.982
7	23.687.517	9.655.318	7.140.089	8.559.008	12.252.126
8	23.003.453	9.072.066	6.590.325	7.996.989	11.594.643
16	17.886.918	5.399.277	2.923.687	4.329.826	7.649.612
24	13.907.607	677.965	15.324	59.765	3.916.259
32	9.664.092	5.877	8.134	12.354	34.427
40	5.852.167	3.618	3.583	4.664	5.028

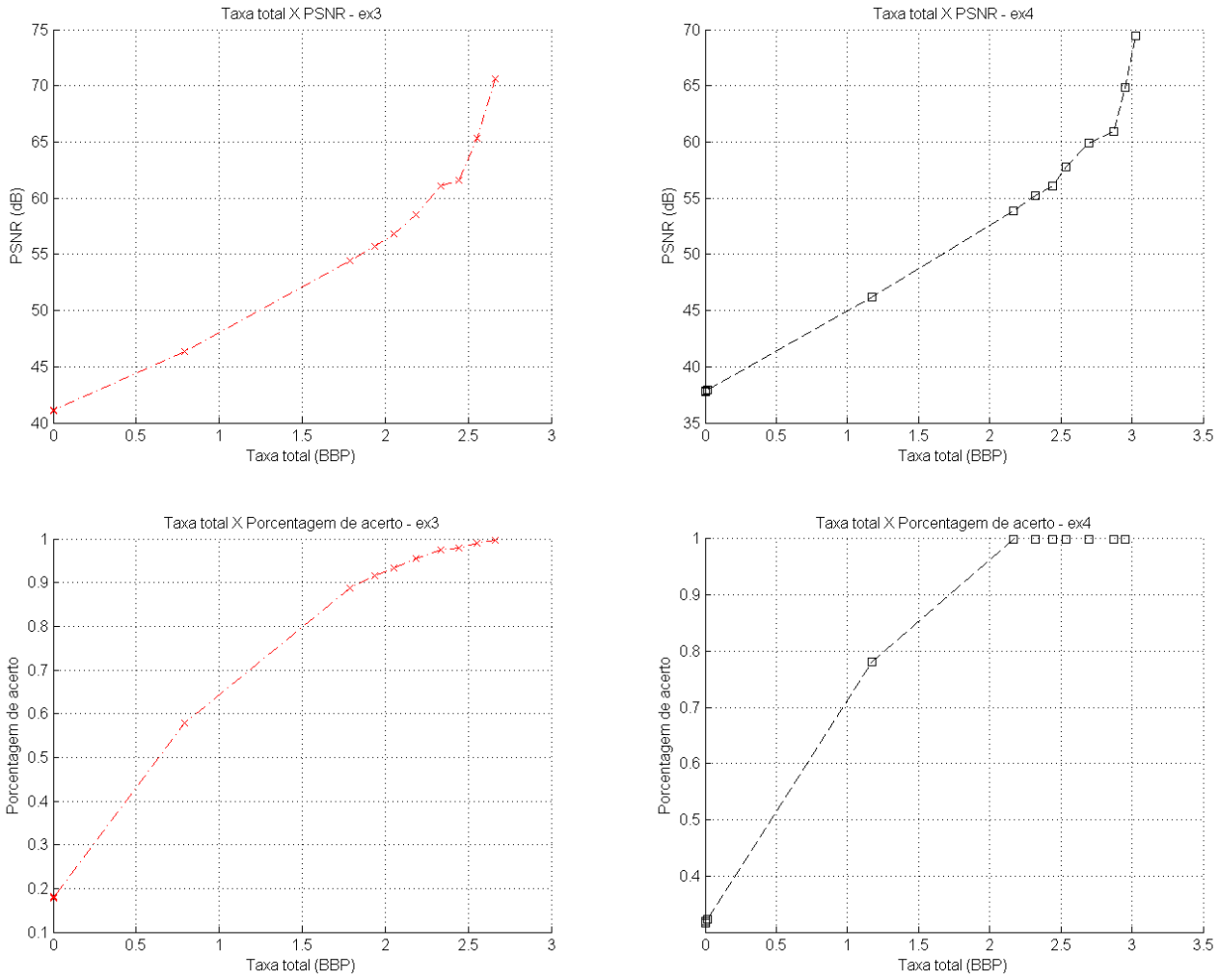


Figura 6.5: Gráficos das abordagens Extensão 3 bits e Extensão 4 bits para NC003071.

Tabela 6.4: Taxa de compressão para o FASTA NC003071.

Abord.	qp 1	qp 2	qp 3	qp 4	qp 5	qp 6	qp 7	qp 8	qp 16	qp 24	qp 32	qp 40
est	0,68	0,71	0,72	0,75	0,79	0,82	0,84	0,87	1,11	1,43	2,06	3,40
etv5	1,62	1,65	1,75	1,83	1,91	1,96	2,06	2,20	3,69	29,39	3390,22	5507,00
ex3	2,03	2,12	2,21	2,32	2,47	2,63	2,79	3,02	6,81	1300,20	2449,51	5560,79
ex4	1,79	1,83	1,88	2,00	2,13	2,21	2,33	2,49	4,60	333,38	1612,78	4271,94
ex5	1,32	1,37	1,41	1,44	1,51	1,58	1,63	1,72	2,60	5,09	578,74	3962,67

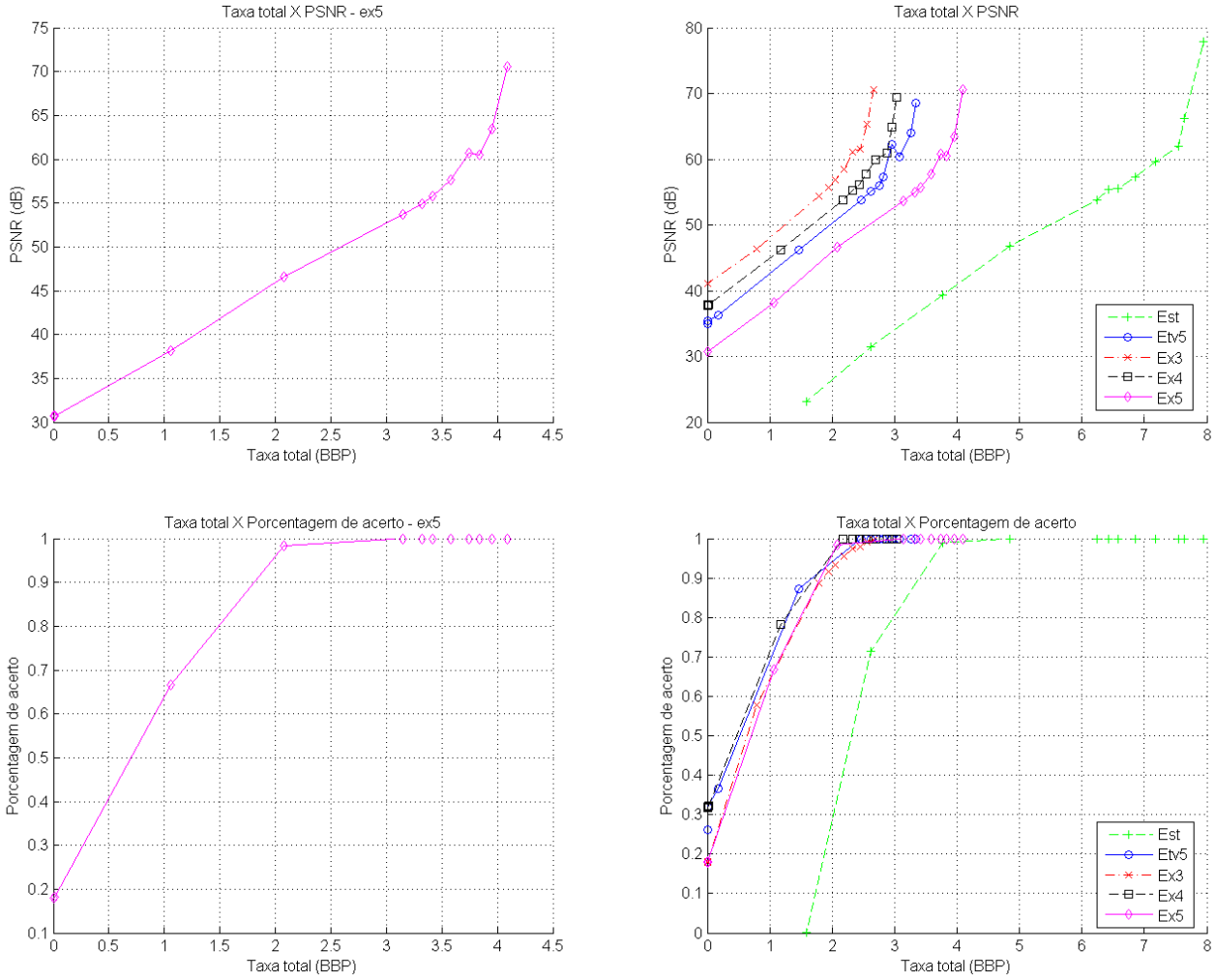


Figura 6.6: Gráficos das abordagens Extensão 5 bits e Todas as abordagens juntas para NC003071.

Taxa compressão do RAR: 3,4302

Entropia do texto: bits 1,9428

Essa sequência ajuda a corroborar as sequências disponíveis anteriormente, pois muitas das considerações ainda se mantém.

A abordagem EST (lado esquerdo da Figura 6.7) manteve a boa precisão de NC_003071 (lado esquerdo da Figura 6.4) inclusive no quantizador 32 (embora com ressalva, já que inseriu um caractere extra, "-"), mas no quantizador 40 a reconstrução continuou horrível 0,04Confirmou totalmente que essa abordagem produz os maiores tamanhos (Tabela 6.5) e as menores taxas de compressão (Tabela 6.6).

A abordagem ETV5 (lado direito da Figura 6.7) teve no quantizador 40 uma precisão em par com NC_003070 (lado direito da Figura 6.1) e melhor que NC_003071 (lado direito da Figura 6.4), o PSNR também seguiu esse padrão. Possivelmente devido aos caracteres extras, podem estar mais próximos da primeira sequência do que da segunda. Isso salienta um problema da abordagem como um todo, em detalhes a abordagem é bem mais instável e imprevisível. O PSNR e a precisão superficialmente estão constantes, mas a taxa de compressão não, como mostrado na Tabela 6.6, onde no quantizador 3 teve 2 trocas de caractere e no quantizador 2 e 4 não teve nenhuma troca. Essencialmente também a partir do quantizador 32 o tamanho do arquivo gerado cai consideravelmente 6.5 e isso ocorreu nas três sequências (Tabelas 6.1 e 6.3) sendo os menores valores ás vezes. Foi a abordagem com o resultado mais dependente do cenário (sequência dada).

A abordagem EX3 (lado esquerdo da Figura 6.8) manteve o mesmo padrão da Figuras 6.2 e 6.5, se mostrando estável e previsível em qualquer cenário. O tamanho (Tabela 6.5) continuou como os menores, com exceção dos quantizadores acima de 32, onde ETV5 se sobressaiu.

A abordagem EX4 (lado direito da Figura 6.8) continuou com os padrões de PSNR e precisão iguais às Figuras 6.2 e 6.5 no lado direito. Sendo a segunda melhor em tamanho 6.5 e precisão, até o quantizador 32, quando ETV5 cresce em desempenho e EX4 fica em terceiro.

A abordagem EX5 (lado esquerdo da Figura 6.9) continuou com os padrões de PSNR e precisão iguais as Figuras 6.3 e 6.6 no lado esquerdo. Sendo a segunda pior em PSNR, tamanho (Tabela 6.5) e taxa de compressão (Tabela 6.6), a precisão está praticamente em par com as outras abordagens.

Na combinação das abordagens, lado direita da Figura 6.9, os gráficos estão iguais às Figuras 6.3 e 6.6 o que é bom, pois mostra que o programa é bem estruturado, não importando a sequência de entrada, os valores de PSNR e precisão vão estar próximos. Os acertos absolutos, na Tabela 6.6 (as células em azul), com exceção do ETV5 e um valor do EST, também estão estáveis. Podemos previr com grau de acerto confiável que EX3 não tem uma reconstrução perfeita, o EX4 tem uma reconstrução perfeita no quantizador 1 e o EX5 tem uma reconstrução perfeita até o quantizador 7 e o EST tem uma reconstrução perfeita pelo menos até o quantizador 8. O ETV5 que não fica confiável nisso pois em NC_003070 (Tabela 6.2) tem dois valores perfeitos, em NC_003071 (Tabela 6.4) não tem nenhum e nessa sequência (Tabela 6.6) tem três valores e com um separado dos outros, o que não tinha acontecido nas outras sequências. Inclusive ao contrario das outras sequências onde o EX4 tinha o melhor valor de taxa de compressão sobre os valores perfeitos, nessa sequência o melhor foi o ETV5 no quantizador 4, com 1,83,

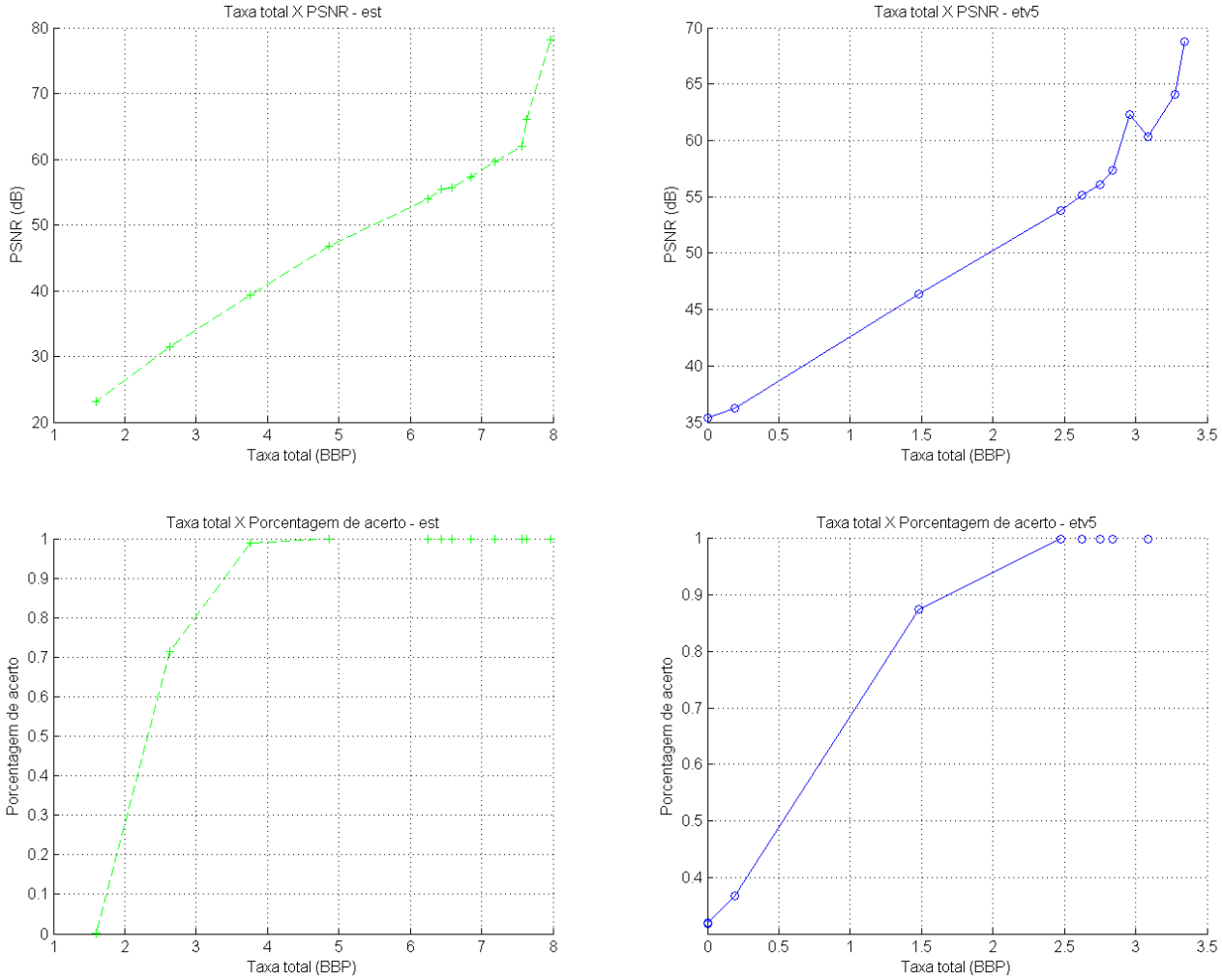


Figura 6.7: Gráficos das abordagens Estável e Estável 5 bits para NC003073.

mas ainda assim abaixo da taxa de compressão do $WinRAR^{®}$. A taxa de compressão do $WinRAR^{®}$ só é alcançada com o quantizador de 16 em diante para ETV5, EX3 e EX4, e de 24 em diante para EX5, não sendo até 40 para EST. O que acaba não sendo proveitoso pois a taxa de erro já está entre 10% e 45%, que para dados sensíveis é muito alto.

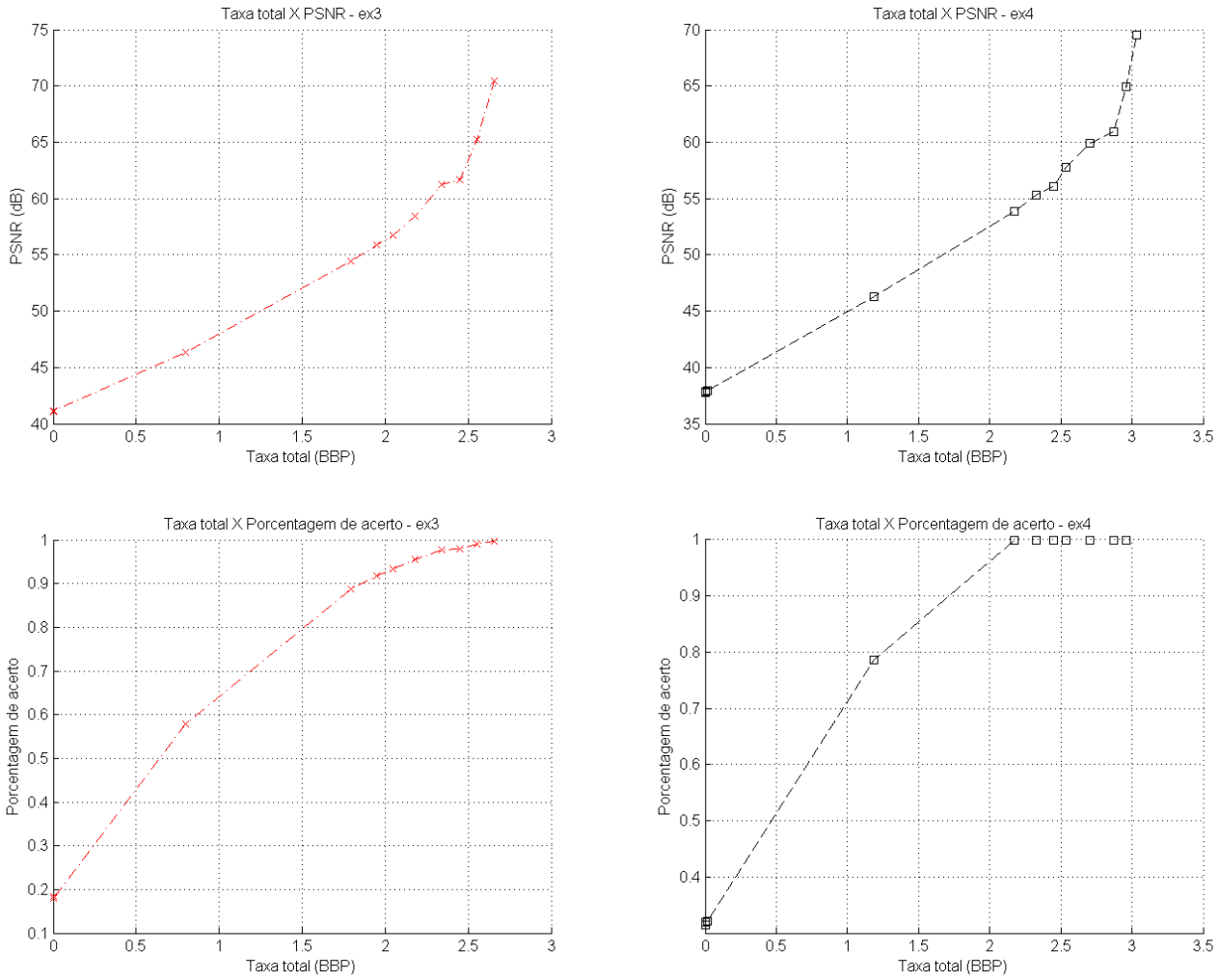


Figura 6.8: Gráficos das abordagens Extensão 3 bits e Extensão 4 bits para NC003073.

Tabela 6.5: Tamanho do H264 junto com arquivo de controle para NC003073.

qp	est	etv5	ex3	ex4	ex5
1	26.213.118	11.005.392	8.760.373	9.983.053	13.466.188
2	25.148.882	10.778.624	8.416.604	9.745.562	13.023.431
3	24.928.467	10.169.108	8.062.676	9.468.403	12.634.190
4	23.680.301	9.749.611	7.714.841	8.897.573	12.313.878
5	22.582.876	9.352.073	7.184.130	8.354.102	11.792.198
6	21.688.385	9.066.999	6.749.453	8.054.353	11.255.662
7	21.178.369	8.640.506	6.418.058	7.668.214	10.967.626
8	20.562.504	8.142.746	5.906.048	7.165.781	10.375.239
16	16.000.869	4.878.078	2.619.528	3.906.185	6.871.011
24	12.376.198	628.266	13.915	53.777	3.497.450
32	8.656.286	5.413	6.892	10.084	32.566
40	5.262.207	3.032	3.089	4.041	4.331

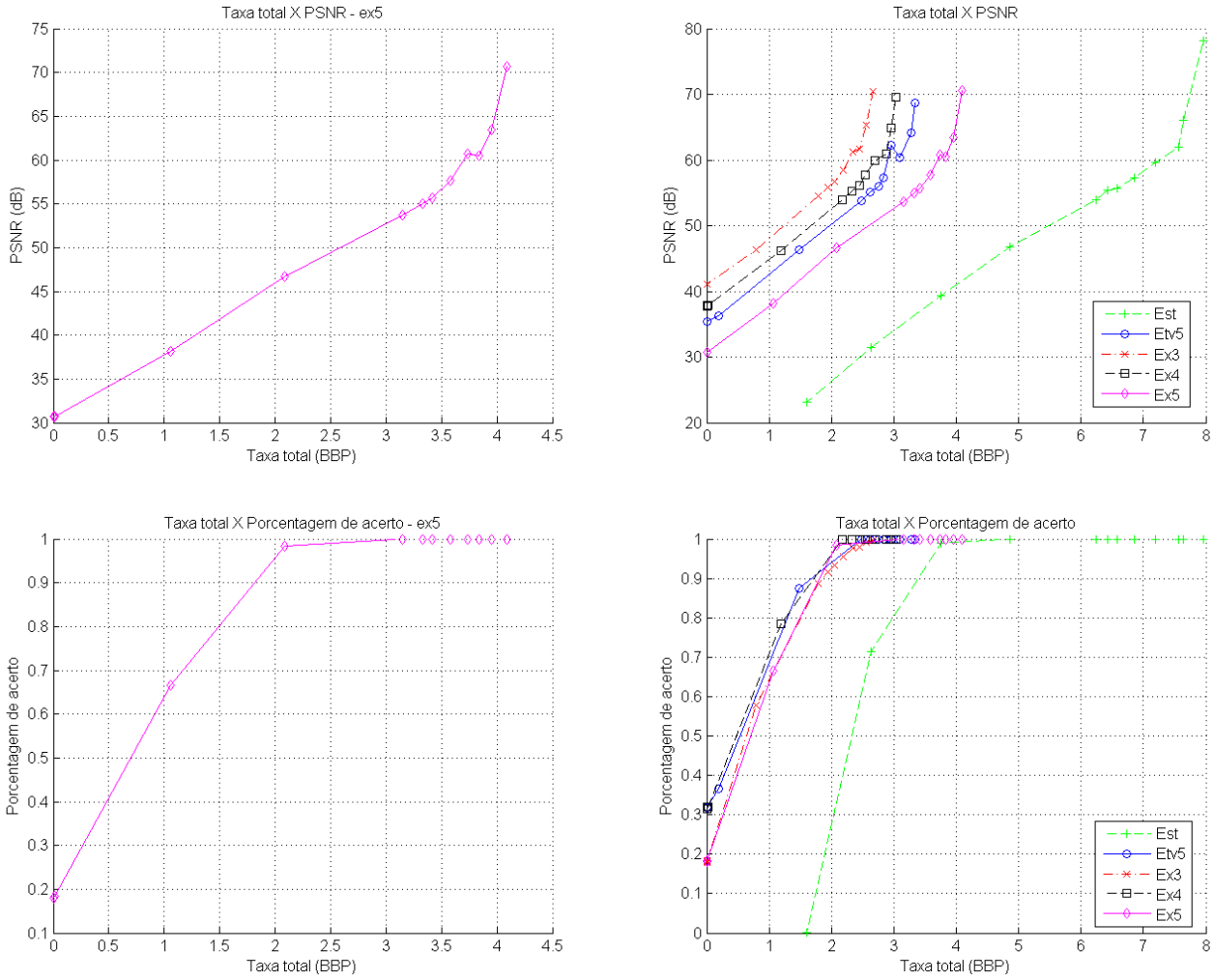


Figura 6.9: Gráficos das abordagens Extensão 5 bits e Todas as abordagens juntas para NC003073.

Tabela 6.6: Taxa de compressão para o FASTA NC003073.

Abord.	qp 1	qp 2	qp 3	qp 4	qp 5	qp 6	qp 7	qp 8	qp 16	qp 24	qp 32	qp 40
est	0,68	0,71	0,71	0,75	0,79	0,82	0,84	0,87	1,11	1,44	2,06	3,38
etv5	1,62	1,65	1,75	1,83	1,90	1,96	2,06	2,19	3,65	28,33	3288,44	5870,82
ex3	2,03	2,11	2,21	2,31	2,48	2,64	2,77	3,01	6,80	1279,22	2582,75	5762,49
ex4	1,78	1,83	1,88	2,00	2,13	2,21	2,32	2,48	4,56	331,00	1765,20	4404,93
ex5	1,32	1,37	1,41	1,45	1,51	1,58	1,62	1,72	2,59	5,09	546,59	4109,98

Capítulo 7

Conclusão

Com o crescente desenvolvimento da tecnologia, os computadores cada vez são mais usados e requeridos nas diferentes áreas da ciência, inclusive nas ciências que não tinha muita tradição, como a biologia. E os computadores agora tem uma parte de destaque, com o mapeamento genético, ao processar as sequências, mas ainda tímido na forma de armazena-las. Além de ajudar a expandir o entendimento da compressão de sequências, esse trabalho tem como objetivo verificar as seguintes hipóteses estabelecidas:

1. É possível comprimir, uma sequência de DNA com um codificador de vídeo com perdas, e poder recuperar a sequência sem erros;
2. Essa compressão é melhor que os métodos usuais de compressão usadas para compactar sequências de DNA (como o $WinRAR^{®}$, Zip ou outros)?;

Sendo desenvolvido um programa que pudesse criar imagens para fazer um vídeo a partir de um arquivo texto e fazer o caminho inverso, com o vídeo obtendo o arquivo texto, de tal forma que se fosse desenvolvido para uso externo só seriam necessários alguns poucos ajustes de interface do usuário. Porém devido ao caráter de estudo, o arquivo era *hardcoded* e o programa executava pelo menos 5 vezes. Foi notado grandemente a influência do tamanho do arquivo no tempo de execução, mas esse é um aspecto não abordado nesse trabalho. Onde os gargalos da execução eram: a criação das imagens (a primeira sequência tinha 114 *frames*, a segunda sequência tinha 75 *frames* e a terceira sequência tinha 67 *frames*), escrever o arquivo reconstruído e comparar os arquivos.

Segundo os testes feitos no Capítulo 6, as abordagens de extensão são as mais estáveis e previsíveis, a Abordagem Estável é a pior possível e embora a Abordagem Estável de 5 *bits* pôde competir com a Abordagem Extensão de 3 *bits* e a Abordagem Extensão de 4 *bits*, a eficiência da abordagem é muito dependente da sequência de entrada. Das abordagens de extensão a melhor, em termos de tamanho de arquivo, taxa de compressão e precisão perfeita, pois as sequências são arquivos de texto que não podem ter erros, foi a Abordagem Extensão de 4 *bits* com quantizador de 1. Pois nas três execuções esse valor tinha uma reconstrução perfeita de um *bitstream* ".264" comprimido e podemos supor que continuará a manter o padrão.

Com os testes podem ter as respostas para as hipóteses feitas:

É possível comprimir, uma sequência de DNA com um codificador de vídeo com perdas, e poder recuperar a sequência sem erros? Sim, é possível comprimir usando técnicas que simulem uma redução de *bits* dos valores.

Essa compressão é melhor que os métodos usuais de compressão usadas para compactar sequências de DNA (como o $WinRAR^{®}$, Zip ou outros)? Pelos testes feitos, infelizmente o algoritmo não é melhor que as técnicas usuais, porque como é um arquivo que tem que ser reconstruído igualmente, não podemos aumentar em demasia o quantizador para reduzir a taxa de compressão, senão induziremos a erros,

Vale salientar que como o teste foi feito em MATLAB, que é uma linguagem interpretada, não podemos definir se o gargalo da criação de vídeo (principalmente) e da escrita do arquivo vai ser diminuído consideravelmente se fosse feito em linguagem compilada, e nem era do escopo básico do programa. Porém se a demora persistir, constituiria de outro ponto contra, pois os métodos mais utilizados são mais rápidos e a rapidez de execução é uma característica que um usuário busca no programa.

Outro ponto a salientar é o aumento do quantizador. Mesmo nas abordagens de extensão o aumento do quantizador para 40 começou a levantar dúvidas, alguns arquivos alertaram para um caractere extra, possivelmente decorrente do valor de termino que está na borda. Mas mostrou que o arquivo tente a ficar extremamente instável, sendo desaconselhado quantizadores maiores.

7.1 Trabalhos futuros

Era esperado que ao termino do trabalho os arquivos pudessem ser comprimidos em um único arquivo, mas o cabeçalho do arquivo original seria perdido, e outros dados também se provaram importantes, até para evitar muitas passagens de dados caso a codificação e a decodificação fossem em computadores diferentes. Mas ainda são dois arquivos e o de controle é um arquivo de texto. E o trabalho está todo baseado no H.264, talvez uma outra codificação possa alcançar resultados melhores. Por esses motivos os próximos passos poderiam ser:

1. Transformar o arquivo de controle em *bitstream*;
2. Os caracteres de extensão podem ter uma compressão para ter mais eficiência;
3. Integrar um codificador de vídeo H.265 (HEVC - *High Efficiency Video Coding*), para poder manipular os *bits* sem pseudo-simular.
4. Verificar a viabilidade de criar blocos repedidos da sequência para trabalhar melhor com os macroblocos e melhorar a codificação H.264.

Referências

[1] Guy E. Blelloch. Introduction to data compression. livro não publicado http://www.cs.duke.edu/courses/spring11/cps296.3/compression.pdf. 1, 13

[2] T.A Brown. *Clonagem gênica e analise de DNA: Uma introdução*. Art Med Editora, 4ª edition, 2003. 1

[3] Andrew Davis. An overview of video compression algorithms. EETimes. 2

[4] Sérgio A. D. Deusdado. *Análise e Compressão de Sequências Genómicas*. PhD thesis, Universidade do Minho - Escola de Engenharia, Departamento de Informática, 2008. 1

[5] Marcio E. Ferreira e Carlos R. B. Neto. A importância da pesquisa genômica e o sequenciamento de DNA, Dezembro 2003. http://www.cenargen.embrapa.br/publica/trabalhos/cot091.pdf. 1

[6] Rafael C. Gonzalez e Richard E. Woods. *Processamento digital de imagens*. Pearson Prentice Hall, 3ª edition, 2010. vii, ix, 2, 12, 13, 17, 18, 19, 20, 22, 23, 24, 25, 26, 29, 30, 36

[7] Michael J. Folk and Bill Zoellick. *File Structures*. Addison-Wesley Publishing Company Inc, 2nd edition edition, 1992. 13

[8] National Center for Biotechnology Information-NCBI. ftp do genbank da arabidopsis thaliana. Website. ftp://ftp.ncbi.nih.gov/genomes/Arabidopsis_thaliana/. 60

[9] Woo-Jin Han Gary J. Sullivan, Jens-Rainer Ohm and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 2012. 37

[10] Bamber Gascoigne. History of Biology. Website HistoryWorld, 2001 ongoing. http://www.historyworld.net/wrldhis/PlainTextHistories.asp?historyid=ac22. 1

[11] Fabiana S. Gonçalves. RNA. Website. http://www.infoescola.com/biologia/rna/. 7

[12] Paulien Hogeweg. The roots of bioinformatics in theoretical biology. *PLoS Comput Biol*, 7(3):e1002021, 03 2011. 1

[13] Markus Hsi-Yang Fritz, Rasko Leinonen, Guy Cochrane, and Ewan Birney. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Research*, 21(5):734–740, May 2011. 1

[14] Equipe knoow.net. Conceito de Ácido Desoxirribonucleico DNA. Website, 2008. http://www.knoow.net/ciencterravida/biologia/acidodesoxirribonucleico.htm. 5

[15] Equipe knoow.net. Conceito de Proteína. Website, 2009. http://www.knoow.net/ciencterravida/biologia/proteina.htm. 3

[16] Ananya Mandal. RNA Types. Website. http://www.news-medical.net/health/RNA-Types.aspx. 7

[17] Silvia Mitiko Nishida Marcelo Santos da Silva. Do que somos feitos. Website. http://www2.ibb.unesp.br/Museu_Escola/4_diversidade/alimentacao/Documentos/2.do_que_somos_feitos.htm. 4

[18] David W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, 2ª edition, 2004. vii, 8, 10, 11

[19] Nomenclature Committee of the International Union of Biochemistry (NC-IUB). *Nomenclature for Incompletely Specified Bases in Nucleic Acid Sequences*, 1984. http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html. 9

[20] Iain E. Richardson. *The H.264 Advanced Video Compression Standard*. Wiley, 2ª edition, 2010. vii, viii, ix, 2, 31, 32, 33, 34, 35, 36, 37, 39, 40, 41, 42, 43, 44

[21] James Rivington. Ultra HD: what you need to know about UHD. Website. http://www.techradar.com/news/home-cinema/high-definition/ultra-hd-what-you-need-to-know-about-uhd-1048954. 1

[22] João C. Setubal and João Meidanis. *Introduction to Computation Molecular Biology*, chapter Basic concepts of molecular biology. PWS Publishing, 1997. 2, 3, 5, 7, 8

[23] Anne Thompson. How James Cameron's Innovative New 3D Tech Created Avatar. Website. http://www.popularmechanics.com/technology/digital/visual-effects/4339455. 1

[24] W. Timothy J. White and Michael D. Hendy. Compressing DNA sequence databases with coil. *BMC Bioinformatics*, 9, 2008. 1